XR16 PRODUCTION PROGRAMMER

USER MANUAL

Revision 2.400

GP Industrial Electronics Ltd,
Unit E,
Huxley Close,
Newnham Industrial Estate,
Plymouth.
PL7 4JN

Tel.: 0752 342961
Fax: 0752 342764

~~Stack~~ Stag Programmer Ltd.
Tel.: 707 332148
Martinfield/Welwyn garden City
Herts
AL7 1JT

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The XR16 is an advanced device programming system utilising a powerful mainframe in conjunction with a series of programming modules. Enabling the system to bulk program a wide range of 24, 28, 32, and 40 pin devices. The mainframe has a standard user accessible RAM of 512k bytes and is controlled by a Z80 micro processor running at 4Mhz.

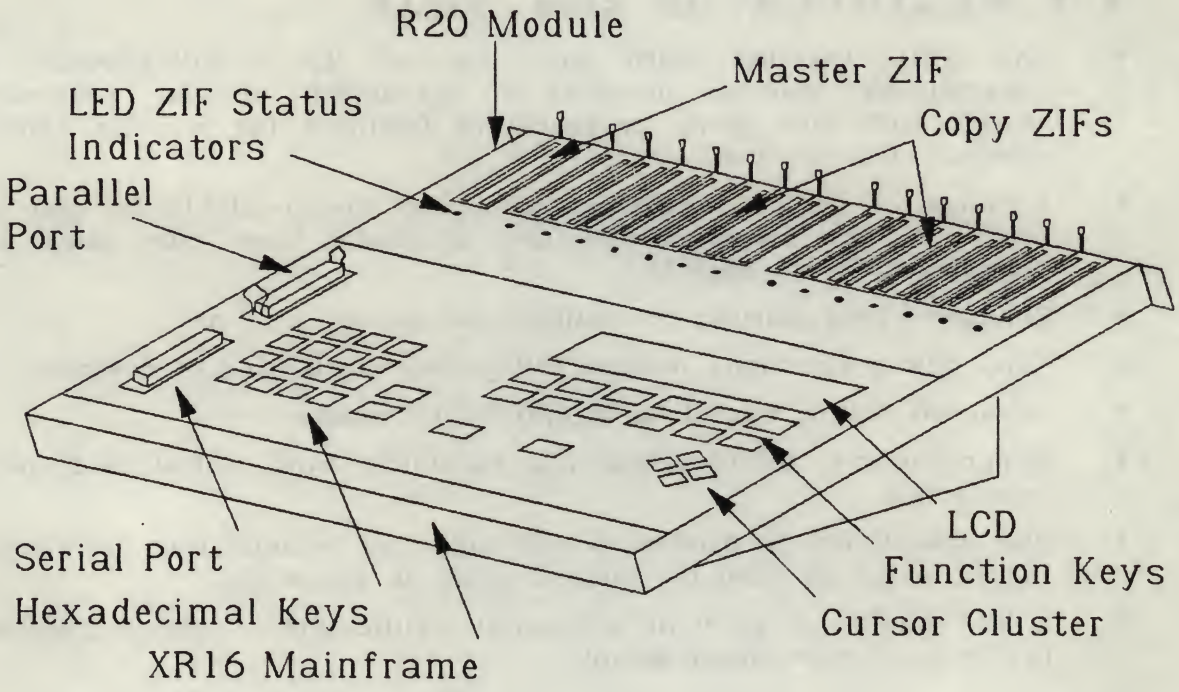
1.1 Features of the XR16

* The XR16 together with any one of its EPROM/EEPROM gang programming modules provides an advanced, modular programming system with full gang programming facilities for a wide range of programmable devices.
* A range of device programming modules are available for the XR16. Detail of the latest range are available from our sales office (Telephone: (0752) 342961).
* Chain/Set programming for multiple device applications.
* Many device functions include full parametric testing of devices.
* Advanced editor, operating in 8/16/32-bit modes.
* Comprehensive bidirectional I/O including both serial and parallel interfacing.
* The operational parameters which are user defined may be stored in the on board EEPROM for instant recall on power up.
* A lock facility to prevent accidental modification of system parameters in the production environment.

2 Getting Started

With the XR16 on the bench in front of you, connected to the mains and with your chosen module in place (see section 3.3 'Insertion/Removal of Modules') the system is ready for operation.

2.1 View of the XR16



2.2 System Power-up

Switch on the XR16 using the mains switch situated at the rear of the machine. Once powered-up the system will display the system start-up page on the LCD (see section 2.1 'View of the XR16'). The XR16 will then perform a brief series of self tests.

XR16 Ver 1.0 (c) GP 13/10/1987 **Initialisation Busy**

System Start-up Page

Should your XR16 unit be found to be faulty, an error message will be displayed (see section 5.4 'Appendix D Error Messages') and system power-up will have failed.

If power-up is attempted before a module has been inserted an error message will be displayed and a tone will sound. Turn off the machine insert the required module (see section 3.3 'Insertion/Removal of Modules') and power-up the system once again.

When there are no faults found during these tests the XR16 will then display a brief specification of the module connected followed by the size of the system RAM space (an example display follows).

```
XR16 Ver 1.0  (c) GP 13/10/1987
        32x16 ZIF Module  512K RAM
```

The XR16 will then wait for a key to be pressed by the user. Once a key has been pressed by the user the main system menu will be displayed.

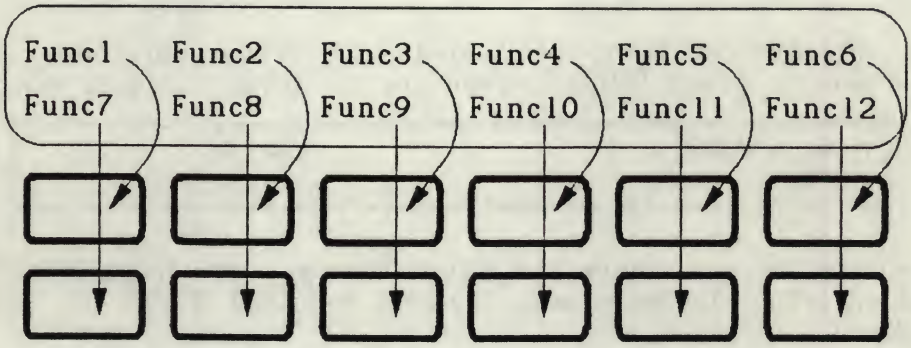
```
In/Out  Remote  Editor
Device  Params  Status  Port  Func  Test
```

Main System Menu

It is from this menu that all XR16 functions may be reached (see section 2.3 'The Menu System').

2.3 The Menu System

The menu system provides an easy and efficient way of selecting functions from the system. A selection from a menu is made by simply pressing the key associated with the desired function. The association of the function keys is explained in the diagram that follows.



LCD to Keyboard Mapping

As it is not possible to implement all of the system functions using only twelve keys many of the menu options lead to sub-menus.

2.4 Example of Operation

It is required that a single Hitachi 27128 device be programmed from a similar preprogrammed device. A possible procedure to carry out this task follows.

After system power-up and the depression of a key by the user, the system main menu will appear on the LCD (see section 2.2 'System Power-up').

In/Out	Remote	Editor			
Device	Params	Status	Port	Func	Test

To select the device type the user must press the **Device** key. The currently selected device, as shown below, is an AMD 2716.

MANUFACTURER	DEVICE
_A.M.D.	2716

Note that the cursor is to the left of the manufacturer entry. Use the **UP/DOWN** cursor keys to change the manufacturer to Hitachi (the manufacturers are in alphabetical order).

MANUFACTURER	DEVICE
_HITACHI	2716

Once Hitachi has been positioned under the manufacturer heading (as shown above) press **RIGHT** cursor key to move to device selection.

MANUFACTURER	DEVICE
HITACHI	_2716

The cursor is now under the device entry. Use the **UP/DOWN** cursor keys to change the device number to the required 27128 (as for manufacturer selection).

MANUFACTURER	DEVICE
HITACHI	_27128

When the required device selection is shown on the display (as above) press the **ENTER** key to activate the device selection.

In/Out	Remote	Editor			
Device	Params	Status	Port	Func	Test

After the **ENTER** key has been pressed the main system menu will be displayed once-again. Now press the **Func** key to enter the device function sub-menu.

Chain	Program	Erase			<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

At this point the source of the data to be programmed must be selected. The currently selected data source is displayed in the top right-hand corner of the LCD. When programming from the master ZIF (as is required in this example) this must indicate **<ROM>.** If it does not do so press the **<RAM>/<ROM>** key to select the required source (see section 4.8.1 'Data Source Selection').

Chain	Program	Erase			<ROM>
CRC	Chksum	Blank	IBC	Verify	Store

When the data source has been defined to be the master ZIF (as shown above) load the master ZIF with the device to be copied, load any of the copy ZIFs with a device to be programmed and press the **Program** key.

Once the program sequence has been completed the status of the programmed device is indicated by the LED ZIF status indicators (see section 5.7 'Appendix G LED ZIF Status Indicators'). Press any key to return to the **Func** sub-menu.

Chain	Program	Erase			<ROM>
CRC	Chksum	Blank	IBC	Verify	Store

Then press the **MENU** key to return to the system main menu.

In/Out	Remote	Editor			
Device	Params	Status	Port	Func	Test

2.5 The MENU Key

The **MENU** key may be used from any menu level to return the user to the top-most system menu.

2.6 The STOP Key

The **STOP** key is used to abort a function during its execution.

Break Command Detected

Once this key has been pressed the above acknowledgement message will appear for a brief period. The user will then be automatically taken to the systems top most menu, as after system power up.

2.7 Entry of Numeric Data

All data is entered into the XR16 using hexadecimal notation. If you are not familiar with hexadecimal notation you should refer to section 5.6 'Appendix F Hexadecimal Notation' covering this subject.

Numeric data is prompted for by a series of dashes on the display. The number of dashes indicating the maximum number of digits which may be entered. It is not necessary to enter leading zeros. In addition if no data is entered a default value of zero will be used.

Data entry is terminated with the **ENTER** key. As each parameter is entered the value is checked for validity. If a parameter is entered which is outside of the permissible range the XR16 will beep and clear that value from the display.

2.7.1 The ENTER Key

The **ENTER** key is used mainly to terminate entry of numeric data. It is used in the same way as the 'Enter' or 'Return' key on a computer. The **ENTER** is at several points also used to make a selection from a list. The use of the key at these points is also as you would expect on a computer.

2.7.2 The CLEAR Key

The **CLEAR** key is used to remove the latest entry in a string of numeric data. It is equivalent to the 'BS' or 'DEL' key on a computer keyboard.

2.8 The Cursor Keys

The four cursor keys are arranged in a diamond pattern. They have one basic function and that is to move the cursor around the LCD display. Whist editing they are used to move the cursor through system RAM to permit viewing/modification of stored data. At other times they are used to scroll the display to facilitate making selections from lists.

3 Operating Considerations

3.1 Mains Supply

The unit will run from either 110V or 240V. It has been factory configured to suit the norm in your locale. The present setting is marked on the back of the unit. Should you need to change this, instructions are given later in the manual.

Under no circumstances should the power supply be adjusted in any way whilst the mains is connected. There are potentially **LETHAL VOLTAGES** present on it.

A mains cable is supplied with the unit complete with a moulded plug.

3.2 Care of the XR16

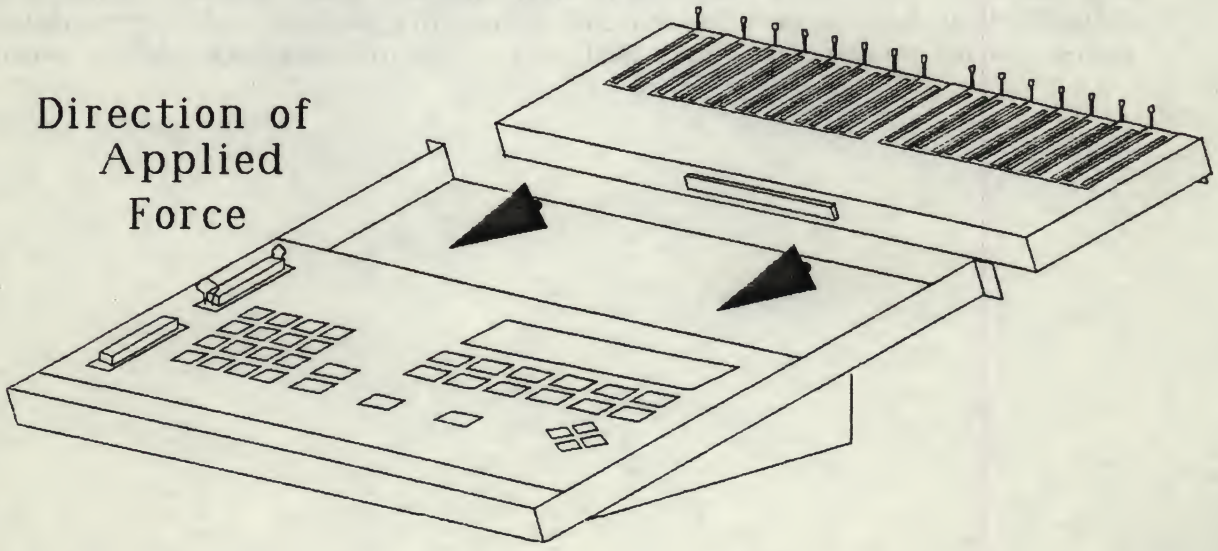
The unit should be cared for in the same way as any other piece of precision electronic equipment. That is it should not be exposed to extremes of temperature or humidity, nor should it be exposed to excessive vibration or shock. The machine will function over a wide range of temperatures from 0°C to 55°C, however many programmable device manufacturers specify that the devices must be programmed at an ambient temperature of 25°C ± 5°C. Certain devices require yet closer temperature tolerances.

The unit may be cleaned with a mild detergent solution and a lint free cloth. It is important that no moisture be allowed to enter the unit. Such cleansing should only be carried out with the unit disconnected from the mains supply.

3.3 Insertion/Removal of Modules

Module Insertion

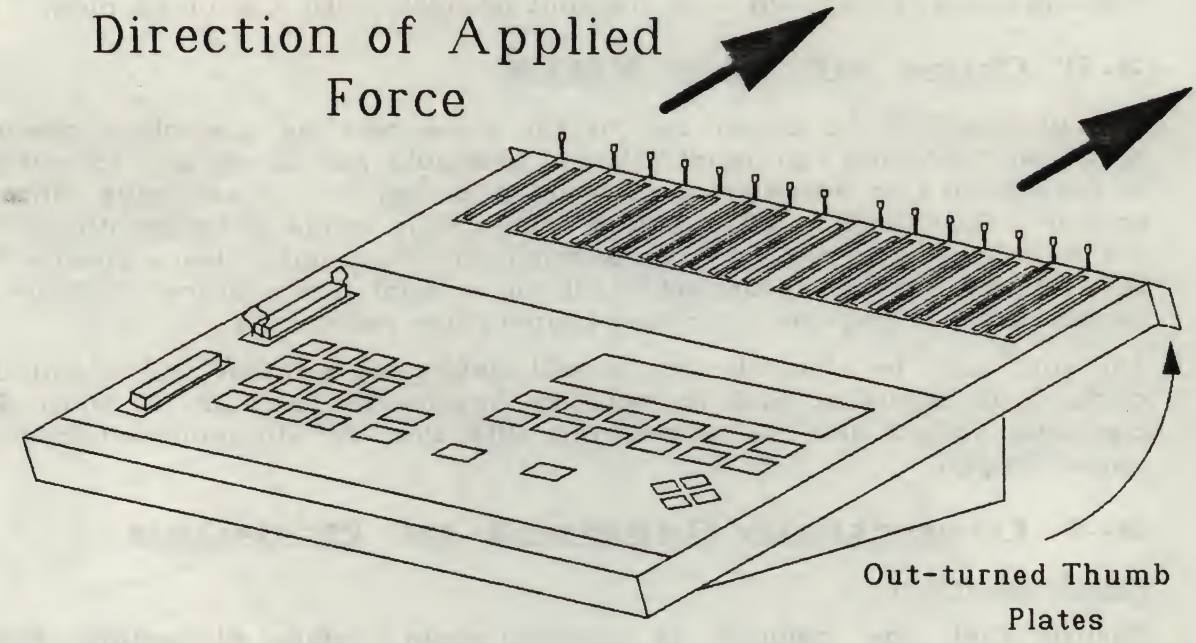
Ensure that the machine is powered-down before attempting module insertion. Then simply align the module with the aperture and push in gently. Force should not be required during insertion of the module. If difficulties are encountered remove the module and inspect the aperture for obstruction and the connectors for damage.



Note:- On no account should excessive force be used to insert the module. If difficulties are encountered using a light pressure inspect the aperture for obstruction and the connectors for damage.

Module Removal

Firstly ensure that the XR16 is powered-down. Then, whilst facing the rear of the system, place thumbs on the out-turned plates at each side of the module. Then place the fore-fingers of each hand behind the down-turned plate running the length of the module. The module may now be drawn out from the XR16 by applying a light pressure with the thumbs and fore-fingers in position.



3.4 Module Connectors

The module connectors are very high quality gold plated connectors. We estimate that with normal usage the connectors should last approximately 20 years. We do however suggest that you avoid unnecessary module removals.

4 The XR16 Command Set

On completion of the power up tests the menu shown below is displayed.

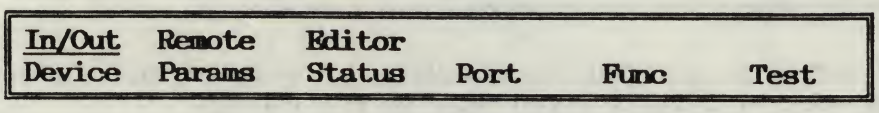
In/Out	Remote	Editor			
Device	Params	Status	Port	Func	Test

The following provides a brief introduction to each of the options. A more detailed explanation of each is given later in the manual.

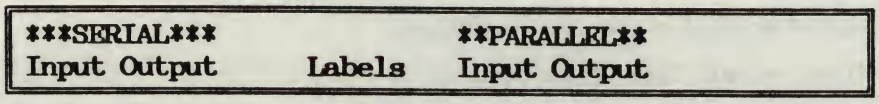
* In/Out - External communication control for data transfer and printing.
* Remote - Activates the remote control mode.
* Editor - System RAM data manipulation.
* Device - Device selection function.
* Params - System parameter manipulation.
* Status - Displays system parameters.
* Port - Communications port manipulation.
* Func - Program device functions.
* Test - System test functions.

4.1 Input/Output Functions

The **In/Out** command is used to select the mode of external communication required of the XR16. Labels for programmed devices may also be designed and/or printed from this function.



Press **In/Out** to enter the input/output selection sub-menu



Upon selection of this function the above sub-menu will be displayed. This then enables the type of physical connection and the direction of data flow to be defined. Label production may also be selected at this menu level.

4.1.1 Serial Input

SERIAL		**PARALLEL**
Input Output	Labels	Input Output

Press **SERIAL Input** from 'In/Out' sub-menu.

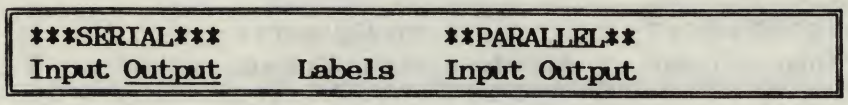
This function may be used to load data into the XR16 via the serial port. The data is expected in the format currently selected (see section 4.7.1.1 'Serial Format').

SERIAL IN
Enter Offset _____

The display then prompts for an offset for the input data. This will be added to the value of the address field, in the data record, to generate the new code location.

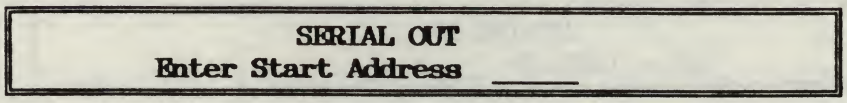
The XR16 will wait for data until either the transfer is completed, an error occurs or the **STOP** key is pressed.

4.1.2 Serial Output

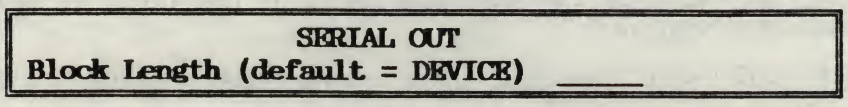


Press **SERIAL Output** from 'In/Out' sub-menu.

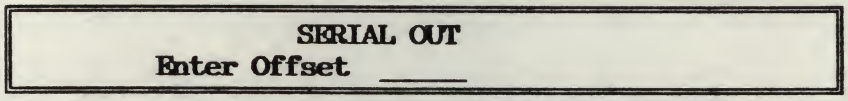
This function allows data to be transmitted from the XR16 via the serial port. The data is transmitted in the format currently selected (see section 4.7.1.1 'Serial Format').



Enter the start address of the data for serial output



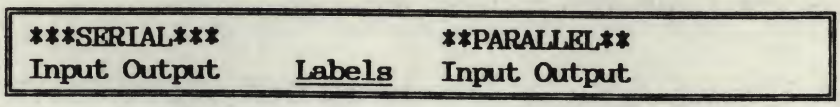
Then enter the length, in bytes, of the block to be transmitted. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered.



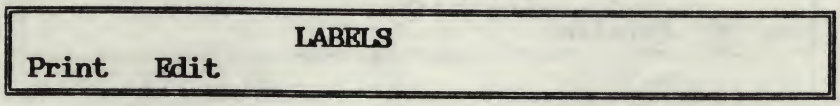
Enter the byte offset of the data to be transmitted. This offset is added to the start address and used in the address field for the first record, with all following address fields being based upon it.

4.1.3 Labels

The XR16 has the facility to print device labels. Data is entered via the keyboard by selecting from a pallet of ASCII characters. The print facility is designed to work with labels supplied by GP and a printer with tractor feed, printing at 10 characters/inch.



Press **Labels** from 'In/Out' sub-menu.



Upon selection of this function the above sub-menu will be displayed. This then enables either print a label or enter a new label to be selected.

4.1.3.1 Printing Labels

```
LABELS
Print  Edit
```

Press **Print** from 'Labels' sub-menu.

This function allows the user to print labels which have been previously entered into the machine. The labels may be printed on a printer with either a serial or a parallel interface.

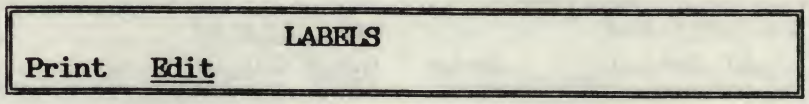
```
PRINT LABELS
Serial  Parallel
```

Select **Serial** or **Parallel** printer interface.

```
PRINT LABELS
Enter number of Rows _____
```

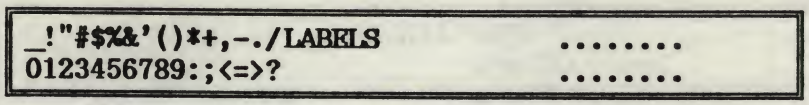
The display then prompts for the number of rows of labels to be printed. This must be entered as a hexadecimal value.

4.1.3.2 Editing Labels



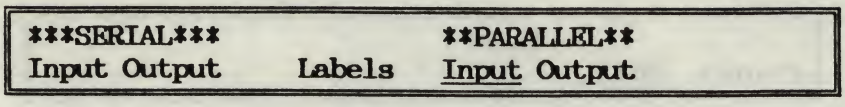
Press **Edit** from 'Labels' sub-menu.

This function enables the user to enter the information which is to be printed on device labels. Selection of characters is made by use of an ASCII pallet. The label information is displayed as it is entered.



Use the cursor key cluster to locate the characters required for the message. Press **ENTER** to select a character and **CLEAR** to delete. The label is displayed as it is constructed on the right-hand side of the LCD. When the message is complete press **MENU.**

4.1.4 Parallel Input



Press **PARALLEL Input** from 'In/Out' sub-menu.

This function is used to load data into the XR16 via the parallel port. The data is expected in the format currently selected (see section 4.7.1.2 'Parallel Format').

The display then prompts for an offset address for the input data. This address must be entered in hexadecimal. A default of zero is used should no value be entered.

The system will wait for the data until either the transfer is completed, an error occurs on the **STOP** key is pressed.

4.1.5 Parallel Output

```
***SERIAL***                    **PARALLEL**
Input Output      Labels    Input Output
```

Press **PARALLEL Output** from 'In/Out' sub-menu.

This function allows data to be transmitted from the XR16 via the parallel port. The data is transmitted in the format currently selected (see section 4.7.1.2 'Parallel Format').

```
PARALLEL OUT
Enter Start Address _____
```

Enter the start address of the data for parallel output

```
PARALLEL OUT
Block Length (default = DEVICE) _____
```

Then enter the length, in bytes, of the block to be transmitted. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered.

```
PARALLEL OUT
Enter Offset _____
```

Then enter the byte offset of the block to be transmitted. This offset is added to the start address and used in the address field for the first record, with all following address fields being based upon it.

4.2 Remote Control Mode

The remote mode enables the XR16 to be operated from a remote work station such as a terminal or PC. It enables all of the functions of the XR16 to be accessed without using the XR16 keyboard.

In/Out Device	Remote Params	Editor Status	Port	Func	Test

Press **Remote** to enter XR16 remote control mode.

Remote transfers all input and output to the serial port. Remote port communication protocol will be as currently defined for the serial port (see section 4.7.1.1 'Serial Format').

The 'Remote' function of the XR16 is designed to be as user friendly as possible. It therefore uses the menu system to support input and output of commands and data. To do this it must be connected to a terminal capable of supporting the ANSI terminal command set. Any PC/MSDOS computer should be able to support this standard. If implementation of the ANSI standard is a problem for your application, please contact our office for advice.

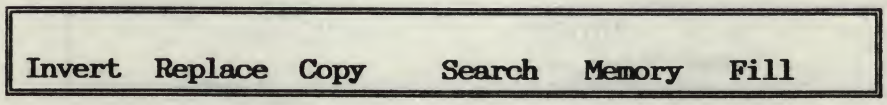
* Cursor control - The diamond formed by the keys 'I', 'J', 'K', and 'M' on an ASCII keyboard is used to move the cursor. These must be upper case characters.
* Command entry - All commands are executed by entering the numeric code associated with the function (as shown in the menu).
* Function Keys - map onto the standard 'QWERTY' keyboard as follows. The **MENU** key may be executed by pressing the '.' key. **CLEAR** maps onto the 'BS' or 'Del' key. The 'ESC' key maps onto the **STOP** key and **ENTER** maps onto the 'Carriage Return' or 'Enter' key on your keyboard.
* Data entry - When operating the memory editor all hexadecimal data should be entered in upper case.
* Terminal configuration - The terminal should be configured with parameters set to match those of the XR16 serial port. Any local echo of characters on the remote terminal should be disabled.

Note:- To mark the end of a data block the <ETX> (03H) character is included during transmission.

4.3 Editor

The editor provides various facilities for the manipulation of data within the RAM of the XR16.

Press **Editor** to enter the 'Editor' sub-menu



The 'Editor' sub-menu is then displayed, as above, from which all editor commands may be selected.

Please note that most editor function commands operate using the byte as the smallest common denominator. However 'Search' and 'Replace' both use the system word (see section 4.5.8 'System Word Length') when referring to search and replace patterns.

The editor has the following facilities:

* Operates in 8/16/32-bit modes.
* Invert - Invert the contents of a memory area (binary NOT function).
* Replace - Search for a byte or series of bytes and replace with a byte or series of bytes.
* Copy - Copy a block of data from one memory area to another.
* Search - Search for a byte or series of bytes in memory.
* Memory - Select a memory address for viewing/modification.
* Fill - Fill a memory area with the data specified.

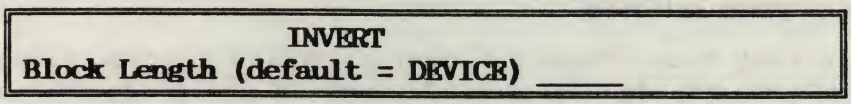
Note:- If the memory has been locked (see section 4.5.4 'Lock Memory') some of the RAM editing facilities will not be made available preventing any unauthorised changes to program data.

4.3.1 Memory Invert

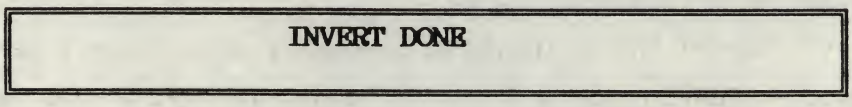
Press **Invert** option from the 'Editor' sub-menu

The **Invert** function will take a user defined block of RAM and perform a binary inversion (binary NOT) of the contents of each byte. Converting all 1's to 0's and 0's to 1's throughout the memory block defined.

Enter start address of the block which you wish to invert.

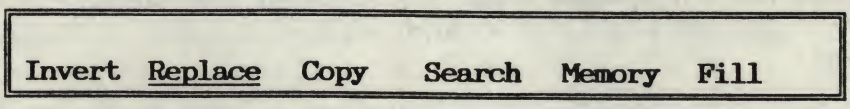


Enter length of block, in bytes, you wish to invert. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.



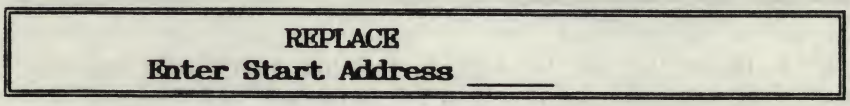
The **Invert** function is completed. Press any key to return to 'Editor' sub-menu.

4.3.2 String Replacement

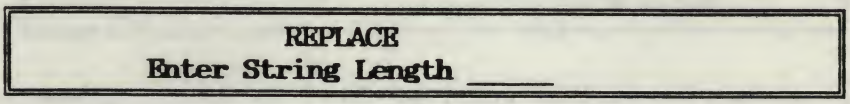


Press **Replace** from the 'Editor' sub-menu.

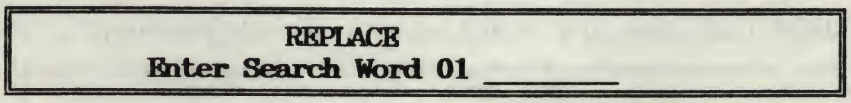
The **Replace** option will search the defined memory area for a pattern, if the pattern is found it will be replaced by the specified replacement.



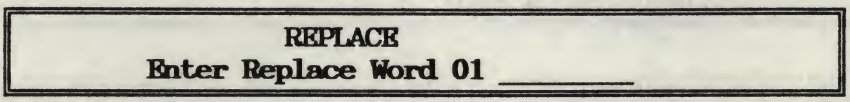
Enter address of start of the area which you wish to search.



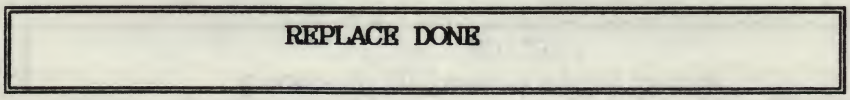
Enter length of string, in system words, you wish to search for (see section 4.5.8 'System Word Length'). This value must be entered in hexadecimal.



Enter the first value you wish to search for. This search pattern is a full system word in length and any leading blanks, remaining after data entry, will be set to the default value zero. This request, for a search word, is repeated until the whole of the search pattern has been entered.



Enter the first value you wish to replace with. The replace pattern is a full system word in length and any leading blanks, remaining after data entry, will be set to the default value zero. This request, for a replace word, is repeated until the full pattern has been entered.



The replace process has been successfully completed. Press any key to return to 'Editor' sub-menu.

If **STOP** is pressed the replace process will be terminated.

Example of String Replacement:

It is required to search from the start of the memory for the pattern '320447AF 8FB42874 A21B64C9' and replace it with 'FFFFFFFF FFFFFFFF A21B64CA'.

At this point the system word must be defined to be 32-bits wide (see sections 4.5.6 'System Status' and 4.5.8 'System Word Length') for this example as it involves the use of 32-bit data.

```
REPLACE
Enter Start Address ____0
```

The search is required to begin at the start of the system memory so zero must be entered as its start address.

```
REPLACE
Enter String Length ____3
```

The string length, in this case, is three as the complete search pattern is three system words long.

```
REPLACE
Enter Search Word 01 320447AF
```

Enter '320447AF' the first word of the search pattern.

```
REPLACE
Enter Search Word 02 8FB42874
```

Enter '8FB42874' the second word of the search pattern.

```
REPLACE
Enter Search Word 03 A21B64C9
```

Enter 'A21B64C9' the third and final word of the search pattern.

```
REPLACE
Enter Replace Word 01 FFFFFFFF
```

Enter 'FFFFFFFF' the first replacement word.

```
REPLACE
Enter Replace Word 02 FFFFFFFF
```

Enter 'FFFFFFFF' the second replacement word.

```
REPLACE
Enter Replace Word 03 A21B64CA
```

Enter 'A21B64CA' the third and final replacement word.

```
REPLACE DONE
```

4.3.3 Memory Copy

```
Invert  Replace  Copy  Search  Memory  Fill
```

Press **Copy** from the 'Editor' sub-menu.

The copy facility allows the user to define a block of RAM data in one section of memory and copy it to another section. This facility is useful for duplicating blocks of data or for moving data blocks around memory.

```
COPY
Enter Source Address _____
```

Enter address of the start of the block you wish to copy.

```
COPY
Enter Destination Address _____
```

Enter destination address for the copied block.

```
COPY
Block Length (default = DEVICE) _____
```

Enter length of data block to be copied (the number of bytes in memory block). The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

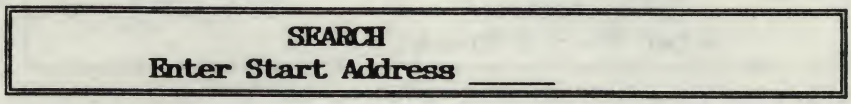
```
COPY DONE
```

The copy operation is completed. Press any key to return to 'Editor' sub-menu.

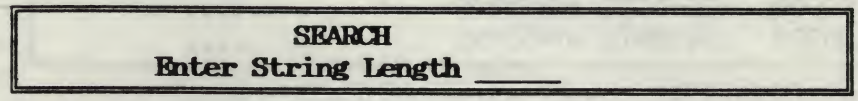
4.3.4 String Search

Press **Search** from the 'Editor' sub-menu.

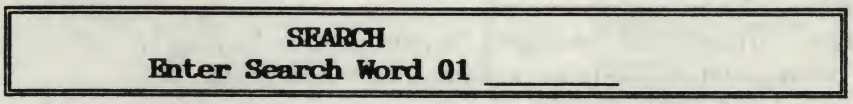
The search facility is provided to locate a string of hexadecimal system words (see section 4.5.8 'System Word Length') stored in the system RAM space.



Enter start address of the memory you wish to search.



Enter length of the search pattern in system words. This value must be entered in hexadecimal.



Enter the first value you wish to search for. The search pattern is a full system word in length. Any leading blanks that remain during data entry will be made the default value zero. This repeats until you have entered all of the string.

The string was not found in the search range. Press any key to return to 'Editor' sub-menu.

The search string was found in the range given. The cursor is positioned at the first byte of the located pattern (as above, when search pattern is 'FFFFFFFF'). Cursor **RIGHT** and cursor **LEFT** keys may be used to then continue the search forward and backward through memory respectively. Press the **MENU** key to return to 'Editor' sub-menu. If **STOP** is pressed the search process will be terminated.

4.3.5 Memory Modification

```
Invert  Replace  Copy  Search  Memory  Fill
```

Press **Memory** from the 'Editor' sub-menu.

This is the memory display/modify function and it enables the user to examine and change the contents of the user memory.

```
MEMORY
Enter Start Address _____
```

Enter the address which you wish to begin view/edit.

```
00000   FFFFFFFF FFFFFFFF        ........
00008   FFFFFFFF FFFFFFFF        ........
```

The data for the specified address is displayed. The row of dots after each row of hexadecimal data represent the ASCII equivalent of the hexadecimal data. (Non-printable characters are shown as full stops (periods)). The number of words of data displayed will change depending upon the defined system word length (the above example shows data with a 32-bit system word defined).

In this function the cursor keys may be used to view the memory at other addresses and new data may be entered at any time by entering hexadecimal data from the keyboard.

Example of Memory Modification:

It is required to modify a two byte (16-bit) area of RAM at memory location 100H. This is say a program version number and it must be updated from '0103' to '0200' (version 1.03 to version 2.0).

To carry out this task it is suggested that the selected system word be defined as 16-bits as in this example (it may also be carried out with an 8-bit word defined).

```
MEMORY
Enter Start Address _____
```

Enter '100' as the edit start address required.

```
00100   0103 FFFF FFFF FFFF      ........
00108   FFFF FFFF FFFF FFFF      ........
```

The required memory location and its present contents will now be displayed and the cursor positioned under this data (as shown above). A number of additional data words will also be displayed (being dependant on the current system word length) following the required data. In this example they are of no interest but may in other instances prove very useful.

Overtype the new data for this location ('0200').

```
00100   0200 FFFF FFFF FFFF      ........
00108   FFFF FFFF FFFF FFFF      ........
```

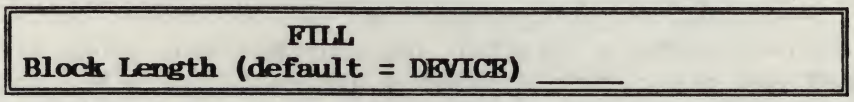
The cursor was incremented as the value was entered and is now located at the start of the second system word displayed on the screen. If an error was made during data entry the **CLEAR** key may now be used to make corrections and/or the cursor keys may be used to explore the surrounding data area. Press the **MENU** key to terminate the memory edit.

4.3.6 Fill Memory

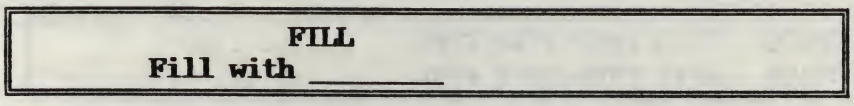
Press **Fill** from the 'Editor' sub-menu.

The fill function provides a facility for the user to define an area of RAM and fill it with a single byte of data.

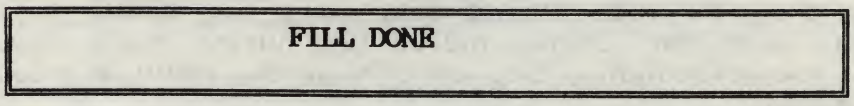
Enter start address of memory block you wish to fill.



Enter length, in bytes, of the memory block you wish to fill. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.



Enter the data that is to be used to fill.



The **FILL** function is completed. Press any key to return to 'Editor' sub-menu.

4.4 Device Selection

The device function is used to select the manufacturer and type of the devices to be used with the XR16. It is essential to select a device type before placing any devices in the sockets of the XR16 module.

In/Out	Remote	Editor			
Device	Params	Status	Port	Func	Test

Press **Device** to select the required device type.

MANUFACTURER	DEVICE
_A.M.D.	2716

The display shows the device selection menu. The device menu displays two headings, 'MANUFACTURER' and 'DEVICE'. Under each heading is one entry from each of the manufacturer and device lists.

The **UP** and **DOWN** cursor keys allow scrolling through the entries under the currently selected heading (indicated by the flashing cursor), while the **LEFT** and **RIGHT** cursor keys move the cursor between the two headings. To make a selection from the device menu press the **ENTER** key on the XR16 keyboard when the desired manufacturer and device are shown on the display.

Example of Device Selection:

It is required to change the selected device to an Hitachi 27128. The currently selected device, as shown below, is an AMD 2716. This may not be true in your instance but the procedure applied to re-selection remains the same.

MANUFACTURER	DEVICE
_A.M.D.	2716

Note that the cursor is to the left of the manufacturer entry. Use the **UP** cursor to change the manufacturer to Hitachi (the manufacturers are in alphabetical order).

MANUFACTURER	DEVICE
_HITACHI	2716

Press **RIGHT** cursor to activate device selection.

MANUFACTURER	DEVICE
HITACHI	_2716

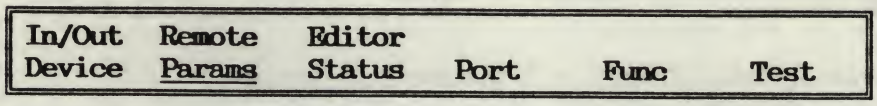
The cursor is now under the device entry. Scroll the display up to show '27128'.

MANUFACTURER	DEVICE
HITACHI	_27128

The required device selection is now shown on the display. Press **ENTER** to confirm the device selection.

4.5 System Parameters

The 'Params' menu allows the user to configure various features of the XR16 system.



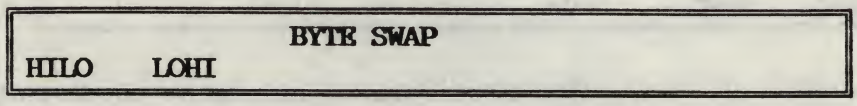
Press **Params** to enter the parameter definition sub-menu.

Upon selection of this option the 'Params' sub-menu is displayed as above. From this sub-menu all of the parameter related functions may be called.

4.5.1 Byte Swap

Press **Bytswp** from the 'Params' sub-menu.

The swap facility enables the user to swap over two consecutive bytes stored within the system RAM space (eg. swap high byte with low byte when dealing with 16-bit data). This function is only operable when using a 40-pin module.



Select the required mode of byte swap.

The contents of the system memory has now been byte swapped.

Note: The memory swap has been carried out but will not appear to have been when viewing/modifing memory.

4.5.2 Save Parameters

Bytswp					
Save	Read	Lock	Status	Ident	Bits

Press **Save** from the 'Params' sub-menu.

This function saves the current XR16 operating parameters in non-volatile memory for recall at a later date. As many as five sets of parameters may be stored at a time. Any one of these can be recalled to re-configure the system using the 'Read' function (see section 4.5.3 'Read Parameters').

Note: that parameter set **One** is always recalled after system power-up.

SAVE PARAMETER SET				
One	Two	Three	Four	Five

Press the required parameter set number.

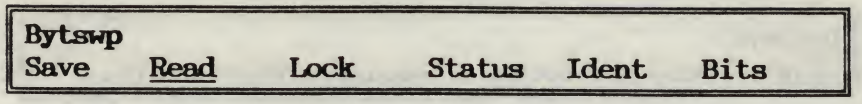
All the parameters are written when the set number is selected, over writing any previous parameters stored in the non-volatile memory under that set number. If the user wishes to recall this operating status after the next system power up, 'Params Read' should be used.

A simple multi-user facility is provided by this function where each of five different users can define their own system configuration. Once defined this may be stored in nonvolatile memory for instant recall at any time.

The parameters which are saved in the non-volatile memory are listed below:

* Device selection (manufacturer and device type).
* Number of Bits selected for programming/editing.
* Automatic device identification enable/disable.
* Byte swap high-low/low-high.
* External Communication Protocols (includes serial baud rate, parity, number of data bits, transmission format, handshaking and number of stop bits).

4.5.3 Read Parameters



Press **Read** from the 'Params' sub-menu.

This function reads into the XR16 operating parameters stored in non-volatile memory.

Press the required parameter set number.

The chosen set of parameters are then read into the XR16 when the set number is selected, over writing any previous parameters.

4.5.4 Lock Memory

```
Bytswp
Save   Read   Lock   Status   Ident   Bits
```

Press **Lock** from the 'Params' sub-menu.

The lock facility is provided so that the machine may be configured for the production environment and locked to prevent accidental modification of the system parameters.

```
System Memory UNLOCKED
Enter Access Code _____
```

Enter the lock code into the system in answer to the prompt.

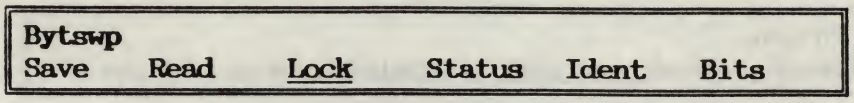
```
System Memory LOCKED
```

The system memory is now locked. Press any key to return to 'Params' sub-menu.

Lock provides the following facilities:

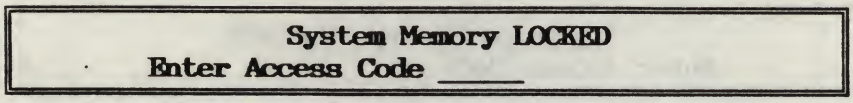
* The system may be locked by a code of up to 5 digits.
* When locked the system prevents the RAM contents being modified.
* When locked system parameters such as device selection can not be modified.
* The system cannot be unlocked without the correct code.
* If the system is powered down the lock condition is cleared (in case the code is forgotten).

4.5.5 Unlock Memory

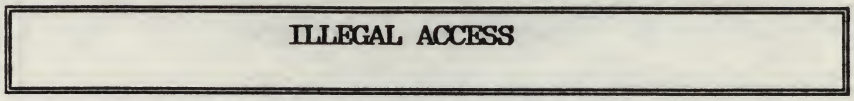


Press **Lock** from the 'Params' sub-menu.

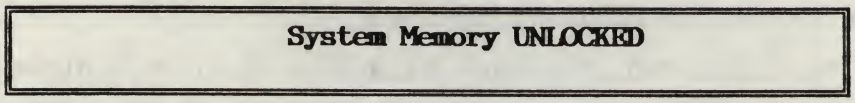
The unlock facility is provided to unlock the system once it has been locked. However this may only be carried out after using the correct access code.



Enter the lock code into the system.

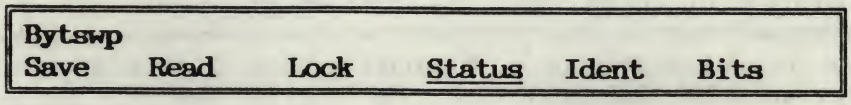


Should the incorrect code be entered the above message will be displayed and a prolonged 'beep' will sound. This may only be terminated by pressing the **STOP** key.



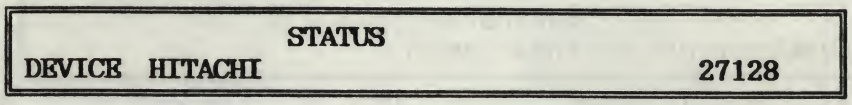
The system memory is now unlocked. Press any key to return to 'Params' sub-menu.

4.5.6 System Status



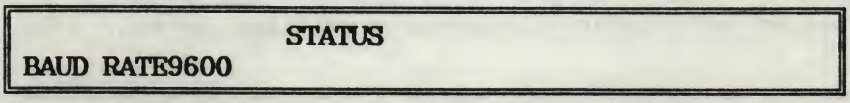
Press **Status** from the 'Params' sub-menu.

The **Status** function allows the system parameters of the XR16 to be viewed.

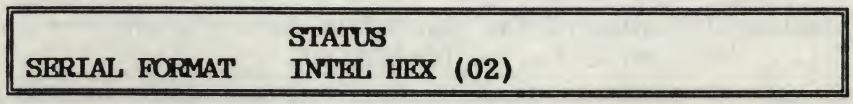


The display shows the first page of the system status, the current device selection (see section 4.4 'Device Selection').

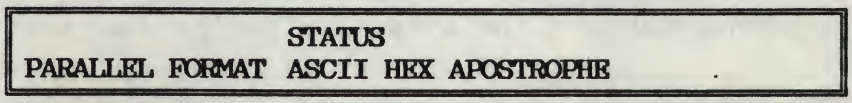
Use of the **UP/DOWN** cursor keys reveals subsequent status pages.



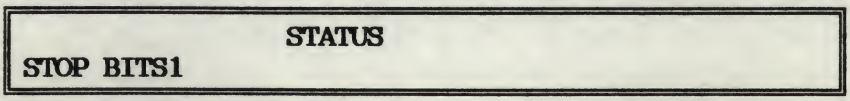
The currently selected serial communication baud rate (see section 4.7.7 'Baud Rate').



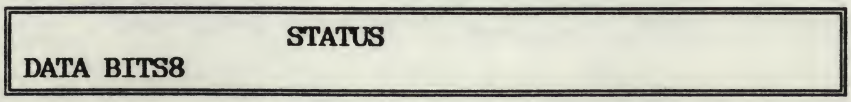
The serial transmission format currently selected (see sections 4.7.1 'Transfer Format' and 5.2 'Appendix B Data Transfer Formats').



The parallel transmission format currently selected (see sections 4.7.1 'Transfer Format' and 5.2 'Appendix B Data Transfer Formats').



The currently selected number of stop bits employed during serial transmission (see section 4.7.5 'Stop Bits').



The selected number of data bits employed during serial transmission (see section 4.7.4 'Data Bits').

```
STATUS
PARITY  DISABLED
```

The operational status of the parity check during serial communication (see section 4.7.6 'Parity').

```
STATUS
HARDWARE HANDSHAKINGDISABLED
```

The operational status of hardware handshaking during serial communication (see section 4.7.2 'Hardware Handshaking').

```
STATUS
XON/XOFF HANDSHAKINGENABLED
```

The operational status of software handshaking during serial transmission (see section 4.7.3 'Software Handshaking').

```
STATUS
BITS / WORD     32
```

The number of bits in a system word (see section 4.5.8 'System Word Length').

```
STATUS
EPROM IDENTIFIER ENABLED
```

The operational status of the automatic device identification facility (see section 4.5.7 'Automatic Device Identification').

```
STATUS
BYTE SWAP:   HILO
```

The operational status of the byte swap facility (see section 4.5.1 'Byte Swap').

The **MENU** key may be used to return to the 'Params' sub-menu at any point within the status pages.

4.5.7 Automatic Device Identification

Press **Ident** from the 'Params' sub-menu.

The more modern of the programmable devices available today have, built into then, a function which enables them to identify themselves to programming systems. The 'Ident' function allows the user to enable/disable the use of this device automatic identification.

Press **Enable** or **Disable** automatic device identification.

Automatic programmable device identification is now set to the operational mode selected.

When enabled the XR16 will attempt to identify devices placed in the copy ZIFs before executing the program sequence. Should any device fail to identify its self as the device selected by the user (see section 4.4 'Device Selection') this failure will be indicated by a flashing LED ZIF status indicator (see section 5.7 'Appendix G LED ZIF Status Indicators'). The XR16 can only succeed to identify a device if that device supports 'Electronic Identification' provided by certain manufacturers. If the device you wish to program does not support this facility or you require the ability to program devices from a mixture of manufacturers then this facility should be disabled.

4.5.8 System Word Length

Bytswp					
Save	Read	Lock	Status	Ident	Bits

Press **Bits** from the 'Params' sub-menu.

The bits option allows the user to define the number of data bits which will be operated on as 1 word when the editor functions are executed as well as during 'Program' and 'Store' operations.

		BITS / WORD
8	16	32

Select the required number of bits/word.

Whilst it is possible to carry out all operations using 8-bit data words, it may be more convenient to use larger data words (ie. when using 16-bit micro processors).

4.6 System Status

The **Status** option allows the user to examine the settings of the XR16 system parameters.

In/Out	Remote	Editor			
Device	Params	Status	Port	Func	Test

Press **Status** to enter the parameter definition sub-menu

STATUS	
DEVICE HITACHI	27128

The display shows the first page of the system status, the current device selection (see section 4.4 'Device Selection').

Use of the **UP/DOWN** cursor keys reveals subsequent status pages.

STATUS
BAUD RATE9600

The currently selected serial communication baud rate (see section 4.7.7 'Baud Rate').

STATUS	
SERIAL FORMAT	INTEL HEX (02)

The serial transmission format currently selected (see sections 4.7.1 'Transfer Format' and 5.2 'Appendix B Data Transfer Formats').

STATUS	
PARALLEL FORMAT	ASCII HEX APOSTROPHE

The parallel transmission format currently selected (see sections 4.7.1 'Transfer Format' and 5.2 'Appendix B Data Transfer Formats').

STATUS
STOP BITS1

The currently selected number of stop bits employed during serial transmission (see section 4.7.5 'Stop Bits').

STATUS
DATA BITS8

The selected number of data bits employed during serial transmission (see section 4.7.4 'Data Bits').

```
STATUS
PARITY   DISABLED
```

The operational status of the parity check during serial communication (see section 4.7.6 'Parity').

```
STATUS
HARDWARE HANDSHAKINGDISABLED
```

The operational status of hardware handshaking during serial communication (see section 4.7.2 'Hardware Handshaking').

```
STATUS
XON/XOFF HANDSHAKINGENABLED
```

The operational status of software handshaking during serial transmission (see section 4.7.3 'Software Handshaking').

```
STATUS
BITS / WORD      32
```

The number of bits in a system word (see section 4.5.8 'System Word Length').

```
STATUS
EPROM IDENTIFIER ENABLED
```

The operational status of the automatic device identification facility (see section 4.5.7 'Automatic Device Identification').

```
STATUS
BYTE SWAP:   HILO
```

The operational status of the byte swap facility (see section 4.5.1 'Byte Swap').

The **MENU** key may be used to return to the 'Params' sub-menu at any point within the status pages.

4.7 Port Protocol

The port function provides a means of defining the operating parameters for external serial and parallel communication.

In/Out Device	Remote Params	Editor Status	Port	Func	Test

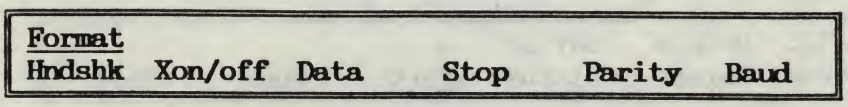
Press **Port** to enter the port definition sub-menu.

Format Hndshk	Xon/off	Data	Stop	Parity	Baud

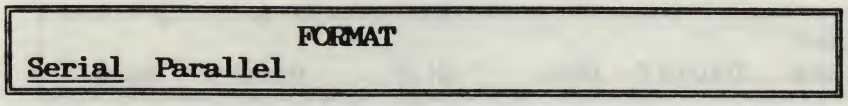
Upon selection of this function the above sub-menu will be displayed. This then enables the serial communication protocols to be defined.

4.7.1 Transfer Format

This function enables the user to select the required data format to be used over the communication ports.



Press **Format** to select data transfer formats sub-menu.



Select physical connection for which the format is to be changed.

For further information on data transfer formats refer to section 5.2 'Appendix B Data Transfer Formats'.

4.7.1.1 Serial Format

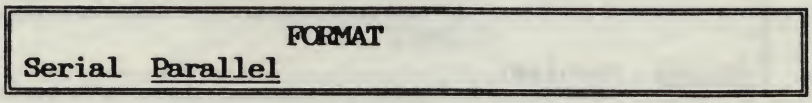
FORMAT
Serial Parallel

Press **Serial** to select serial data transfer formats sub-menu.

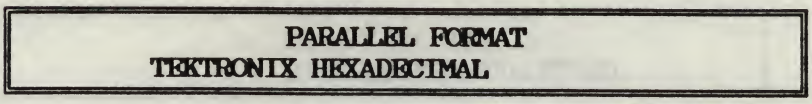
SERIAL FORMAT
TEKTRONIX HEXADECIMAL

The cursor **UP** and **DOWN** keys may be used to scroll through the list of possible serial data formats. When the required format is displayed on the screen press **ENTER** to select (see section 5.2 'Appendix B Data Transfer Formats').

4.7.1.2 Parallel Format



Press **Parallel** to select parallel transfer formats sub-menu.



The cursor **UP** and **DOWN** keys may be used to scroll through the list of possible serial data formats. When the required format is displayed on the screen press **ENTER** to select. (see section 5.2 'Appendix B Data Transfer Formats').

4.7.2 Hardware Handshaking

Format					
Hndshk	Xon/off	Data	Stop	Parity	Baud

Press **Hndshk** to select hardware handshake control.

HARDWARE HANDSHAKING	
Enable	Disable

Press **Enable** or **Disable** of hardware handshake.

Hardware handshaking is now set to the operational mode selected.

Please note it is suggested that hardware and software handshaking are not used together. It is also suggested that for systems which do not support hardware handshaking this option should be disabled.

For further details of handshaking control you should refer to section 5.1.6.1 'Hardware Handshaking'.

4.7.3 Software Handshaking

Format					
Hndshk	Xon/off	Data	Stop	Parity	Baud

Press **Xon/off** to select software handshake control.

XON/XOFF HANDSHAKING	
Enable	Disable

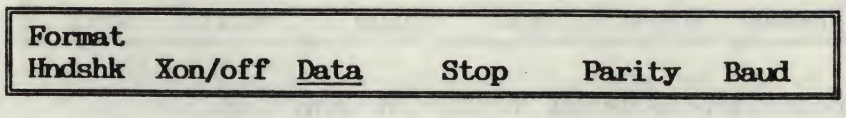
Press **Enable** or **Disable** of software handshake.

Software handshaking will now be set to the operational mode selected.

Please note it is suggested that hardware and software handshaking are not used together. It is also suggested that for systems which do not support hardware handshaking this option should be disabled.

For further details of handshaking modes you should refer to section 5.1.6.2 'Software Handshaking'.

4.7.4 Data Bits



Press **Data** to select number of data bits.

The number of data bits for the serial channel may be modified. A choice of 7 or 8 data bits is offered, selection is made by pressing the appropriate function key.

Press appropriate key to select number of data bits required.

For further information on data bits you should refer to section 5.1.4 'Word Format'.

4.7.5 Stop Bits

Format					
Hndshk	Xon/off	Data	Stop	Parity	Baud

Press **Stop** to select number of stop bits.

The number of stop bits for the serial channel may be modified. A choice of 1 or 2 stop bits is offered, selection is made by pressing the appropriate function key.

STOP BITS	
One	Two

Press appropriate key to select number of stop bits.

The serial port will mow operate using the selected number of stop bits.

For further information on stop bits you should refer to section 5.1.4 'Word Format'.

4.7.6 Parity

Format					
Hndshk	Xon/off	Data	Stop	Parity	Baud

Press **Parity** to select the type of parity check required.

The parity options available are **Even, Odd** and **Disable.** Selection is made by pressing the appropriate function key.

		PARITY
Even	Odd	Disable

Press appropriate key to select parity operating mode.

The serial port will now operate using the selected mode of parity.

For further information on parity options you should refer to section 5.1.5 'Parity'.

4.7.7 Baud Rate

Format Hndshk	Xon/off	Data	Stop	Parity	Baud

Press **Baud** to select the baud rate required.

The Baud Rate of the communications channel may be modified using this function.

19200	9600		BAUD RATE		
4800	2400	1200	600	300	150

Press relevant function key for required baud rate.

Communications will now be carried out at the baud rate selected.

4.8 Device Functions

These functions offer the user various means to program and test devices. Also included is a function to enable the user to define the source of the data to be programmed by the XR16.

In/Out	Remote	Editor			
Device	Params	Status	Port	Func	Test

Press **Func** to enter the device function sub-menu.

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

As this option is selected the above sub-menu will be displayed. The **Erase** option will only be displayed when an EEPROM device has been selected using 'Device' from the main menu (see section 4.4 'Device Selection').

4.8.1 Data Source Selection

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **<RAM>/<ROM>** to select the programming source data.

The programming data source function enables the user to define that either the system RAM or the master ZIF device is to be the source when programming. The display shows which data source is currently selected.

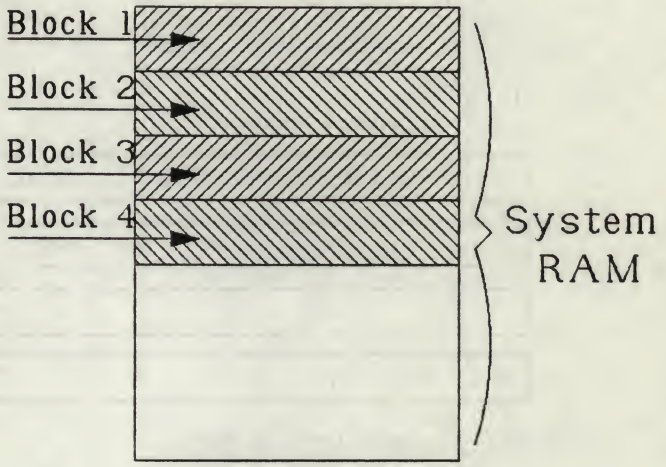
4.8.2 Chain Programming

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **Chain** to execute chain programming.

This function provides a device programming facility for sets of devices on the XR16. Devices are programmed, in sets, with data from the system RAM.

The data must be stored as a sequence of, continuous, blocks in the system RAM space (see diagram). Each block being equal in length to all other blocks and containing the data for a single device in the set. The diagram shows an example with 4 devices/set and the data for each device stored sequentially in the system RAM (blocks 1, 2, 3 and 4 are the data areas occupied by devices 1, 2, 3, and 4 respectively).



Device Memory Map

Data may be loaded from the master ZIF or via a communications port (see sections 4.8.11 'Store' and 4.1 'Input/Output Functions'). The number of sets and/or size of devices may be limited by the size of the your system RAM.

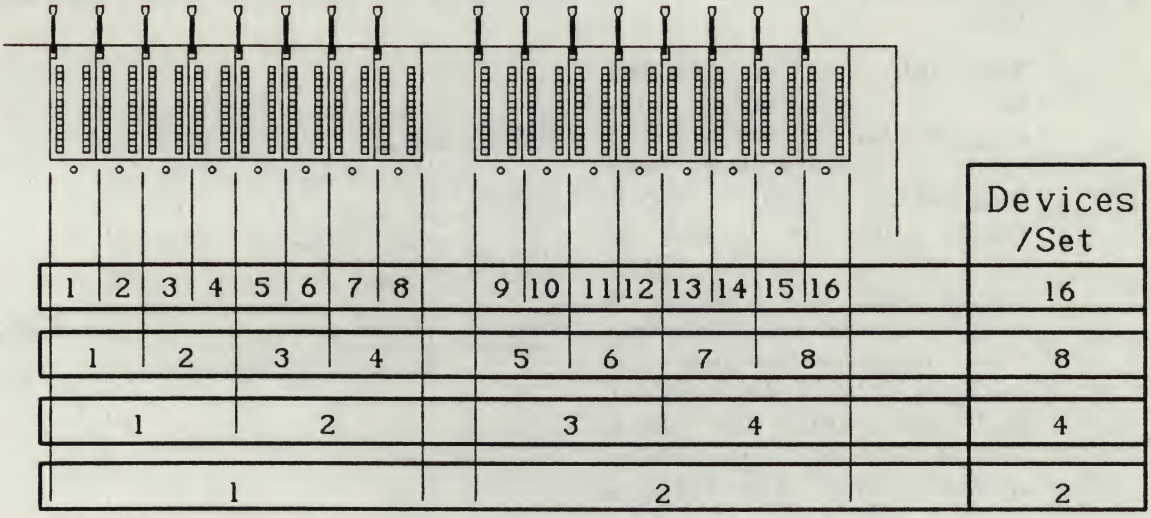
CHAIN PROGRAMMING ERROR
Chain Program in 8 bit mode only

Should chain programming be attempted with the system word length not defined as 8-bits the above error will be reported. Chain programming will only operate with an 8-bit system word and its size must be re-defined (see section 4.5.8 'System Word Length').

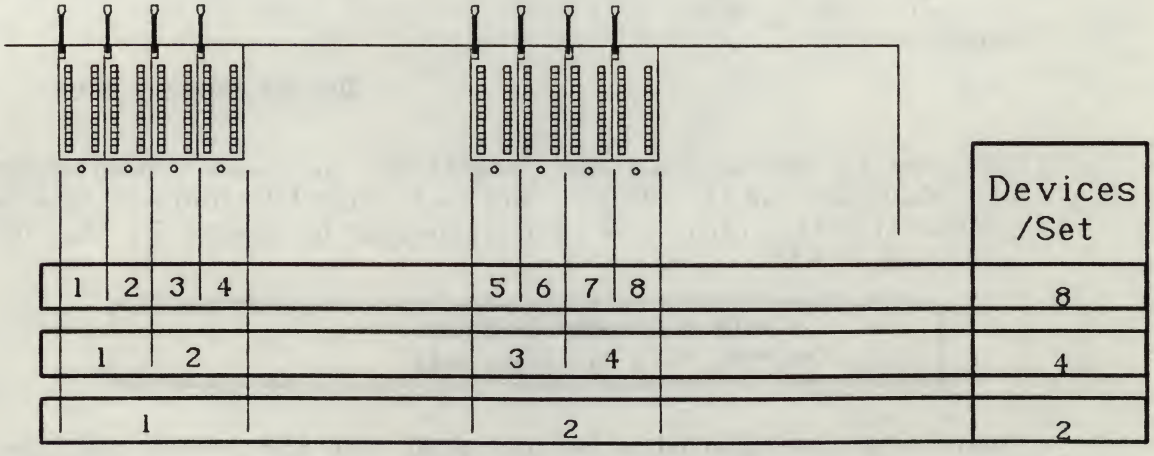
CHAIN PROGRAMMING
Enter Number of Devices / Set __

The required copy ZIFs should be filled before this value is entered into the system. The diagrams below indicate the location of each of the sets once programmed and so may be used to establish which of the copy ZIFs you need load with devices. All modules follow the same logical pattern regardless of the number and size of the ZIFs.

16-ZIF Module Device Map



8-ZIF Module Device Map



Enter the required set size, the number of devices in a programmed set. This size may only be 2, 4, 8, or 16 entered as a hexadecimal value (eg. 2, 4, 8, 10). Clearly however selecting a 16 device set is unwise if you only have an 8-ZIF module.

```
CHAIN PROGRAMMING TEST
024
```

The system will then proceed to illegal bit check each set in turn.

```
CHAIN PROGRAMMING ERROR
Illegal Bit Check Failed
```

When incorrect devices are present in any of the copy ZIFs the test sequence will pin-point them by flashing the relevant LED ZIF status indicators (see section 5.7 'Appendix G LED ZIF Status Indicators').

```
CHAIN PROGRAMMING BUSY
072
```

Each of the device sets will then be programmed in turn. After the program passes a verify pass for each of the device sets will follow.

```
CHAIN PROGRAMMING DONE
```

Any devices failing the verify test pass will be indicated by flashing LED ZIF indicators (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.3 Program Device

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **Program** to program devices in copy ZIFs.

This function provides a device programming facility on the XR16. Devices may be programmed with data from either the master socket or the RAM depending upon the source selected (see section 4.8.1 'Data Source Selection').

Programming can be executed in either 8-bit (normal programming) mode, or 16/32-bit (set programming) mode.

4.8.3.1 8-Bit Programming from Master ZIF

PROGRAM TEST 024

The system will then proceed to illegal bit all devices in the copy ZIFs.

PROGRAM ERROR Illegal Bit Check Failed

When incorrect devices are present in any of the copy ZIFs the test sequence will pin-point them by flashing the relevant LED ZIF status indicators (see section 5.7 'Appendix G LED ZIF Status Indicators').

PROGRAM BUSY

The display then informs the user that the system is busy programming.

4.8.3.2 8-Bit Programming from RAM

```
PROGRAM BUSY
Enter RAM Start Address _____
```

The display prompts for the start address of the RAM block to be used.

```
PROGRAM BUSY
Enter ROM Start Address _____
```

The display then prompts for the ROM address at which the data will be positioned.

```
PROGRAM BUSY
Block Length (default = DEVICE) _____
```

The display then prompts for the size of the block to be programmed, in bytes. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

```
PROGRAM TEST
024
```

The system will then proceed to illegal bit all devices in the copy ZIFs.

```
PROGRAM ERROR
Illegal Bit Check Failed
```

When incorrect devices are present in any of the copy ZIFs the test sequence will pin-point them by flashing the relevant LED ZIF status indicators (see section 5.7 'Appendix G LED ZIF Status Indicators').

```
PROGRAM BUSY
```

The display then informs the user that the system is busy programming.

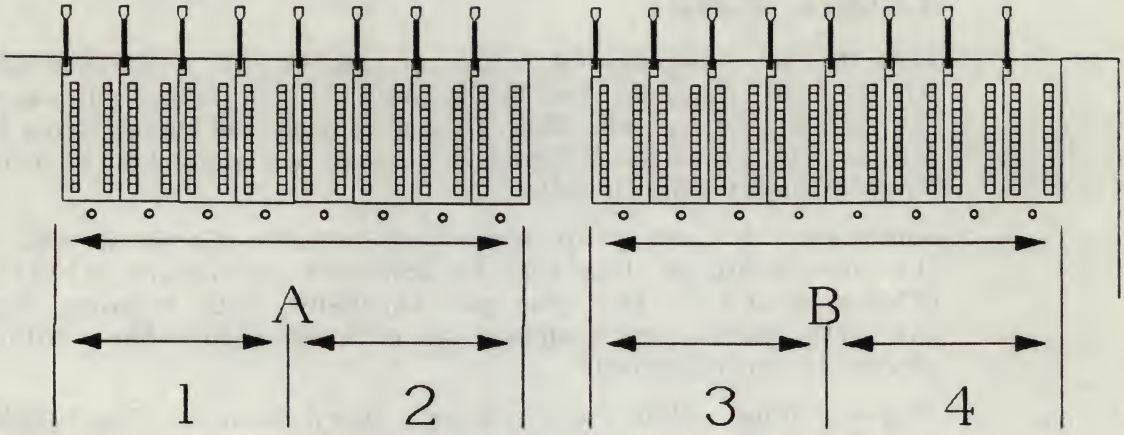
4.8.3.3 16/32-Bit Set Programming from RAM

Data to be programmed must be stored as a stream of interlaced bytes (see diagram '16/32-Bit Data Memory Map' in section 4.8.11 'Store') in the system RAM. This data may be loaded from the master ZIF or via a communications port (see sections 4.8.11 'Store' and 4.1 'Input/Output Functions').

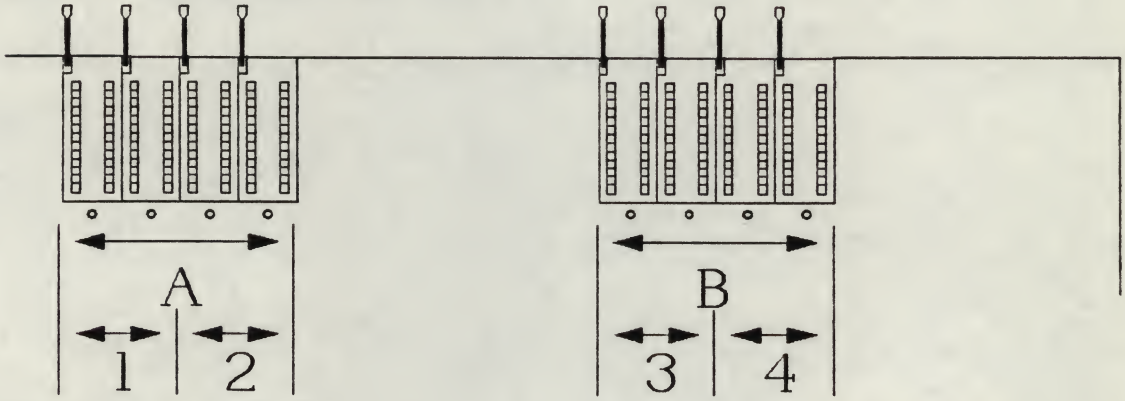
When loaded from preprogrammed devices placed in the master ZIF the interlacing of data may be achieved by use of a byte offset. The offsets stated in the diagram '16/32-Bit Data Memory Map' refer to the byte offset which should be entered before the contents of each device is down-loaded.

Please note that any data down-loaded, for 16/32-bit set programming, via an external communications port must already be interlaced in the manner discussed.

16-ZIF Module Device Map



8-ZIF Module Device Map



16-Bit Programming Key	
A	**B**
High Byte	Low Byte
(Even Address Boundary)	(Odd Address Boundary)
Byte Offset=0	Byte Offset=1

32-Bit Programming Key			
1	**2**	**3**	**4**
Upper-High Byte	Upper-Low Byte	Lower-High Byte	Lower-Low Byte
Byte Offset=0	Byte Offset=1	Byte Offset=2	Byte Offset=3

```
PROGRAM BUSY
Enter RAM Start Address _____
```

The display prompts for the start address of the RAM block to be used.

```
PROGRAM BUSY
Enter ROM Start Address _____
```

The display then prompts for the ROM address at which the data will be positioned.

```
PROGRAM BUSY
Block Length (default = DEVICE) _____
```

The display then prompts for the size of the block to be programmed, in bytes. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

```
PROGRAM TEST
024
```

The system will then proceed to illegal bit all devices in the copy ZIFs.

```
PROGRAM ERROR
Illegal Bit Check Failed
```

When incorrect devices are present in any of the copy ZIFs the test sequence will pin-point them by flashing the relevant LED ZIF status indicators (see section 5.7 'Appendix G LED ZIF Status Indicators').

```
PROGRAM BUSY
```

The display then informs the user that the system is busy programming.

Example of Set Programming:

It is required that a set of four Hitachi 27128 devices (32-bit data set) be programmed with data stored on a preprogrammed set of Hitachi 27128 devices.

In this example the source of the data is a set of similar devices. However data may be down-loaded from dissimilar devices or, via the external communication link, from some other remote equipment.

To carry out this task several of the system parameters must be set. A 32-bit system word length (see section 4.5.8 'System Word Length'), the correct device type (see section 4.4 'Device Selection') and the programming data source must be selected as **<RAM>** (see section 4.8.1 'Data Source selection'). Then data stored on the preprogrammed set must be down-loaded into the system RAM (see section 4.8.11 'Store'). It is suggested that the data be stored from RAM start (address zero) as a large part of the system RAM will be required for the down-load of the four devices. Once all this has been carried out the 'Program' function may be entered and the following carried out.

```
PROGRAM BUSY
Enter RAM Start Address ____0
```

Enter the system RAM location at which the programming data was stored, in this case zero.

```
PROGRAM BUSY
Enter ROM Start Address ____0
```

Enter zero as the whole device is required to be programmed.

```
PROGRAM BUSY
Block Length (default = DEVICE) ____0
```

Load a single device in each of the four ZIF groups **1, 2, 3,** and **4** (see diagram of '8/16-ZIF Module Device Map') and again enter zero (making use of the default).

```
PROGRAM TEST
024
```

The system will then proceed to illegal bit all devices in the copy ZIFs.

```
PROGRAM BUSY
```

The devices have now been programmed and will be an exact copy of the source devices

4.8.4 Visual Verify

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **VisVfy** to visually verify selected source.

This function enables the user to verify data from the selected source (see section 4.8.1 'Data Source Selection') and view any errors discovered as a result. The format of the view will depend upon the module currently inserted into the XR16 system.

Note:- The descriptions made of this function apply to the 16-ZIF modules available. For those modules with 8-ZIFs the function differs only by the number of display pages required to output the verify information.

4.8.4.1 Visual Verify using RAM

```
VERIFY BUSY
03E
```

The function then verifies the data stored in the copy ZIFs indicating the progress of the check by use of a memory block counter.

```
00040 FF FF FF FF XX XX XX XX     <RAM>
      FF FE FF 1A XX XX XX XX  Zifs 1-8
```

If the verify test discovers any errors the display will show these as the first page of the visual verify output. The address of the current sample is displayed in the top left-hand corner of this display. The address is followed by either four or eight hexadecimal values (the quantity depending on the size of the module ZIF sockets, for modules having 40-pin ZIFs this will be four and for modules with smaller ZIFs, eight). The selected data source is shown in the top right-hand corner of the display and below it the numbered range of the ZIF status currently displayed. The bottom row of hexadecimal values displayed to the left of this indicates the actual values found in the tested devices. The presence of a group of 'X's in both rows indicates that no device has been found in this ZIF socket.

```
00040 FF FF FF FF FF XX FF FF     <RAM>
      FF FF FF FF FF XX FF FF  Zifs 9-16
```

The **UP/DOWN** cursor keys may be used to reveal previous/later pages and the **LEFT/RIGHT** keys used to move the display across the array of copy ZIFs. The **MENU** key may be used to return to the 'Func' sub-menu.

```
VERIFY DONE
Hit any Key to Continue
```

Note:- The LED ZIF status indicators do not in this instance indicate the success or failure of the tested devices.

4.8.4.2 Visual Verify using Master ZIF

```
VERIFY BUSY
03E
```

The function then verifies the data stored in the copy ZIFs indicating the progress of the check by use of a memory block counter.

```
VERIFY ERROR
Master ZIF Empty
```

If the visual verify test is attempted without first inserting the required master device in the master ZIF socket the above error will be reported. Press any key to return to the 'Funcs' sub-menu.

```
00040 FF FF FF FF XX XX XX XX     <ROM>
      FF FE FF 1A XX XX XX XX  Zifs 1-8
```

If the verify test discovers any errors the display will show these as the first page of the visual verify output. The address of the current sample is displayed in the top left-hand corner of this display. The address is followed by either four or eight hexadecimal values (the quantity depending on the size of the module ZIF sockets, for modules having 40-pin ZIFs this will be four and for modules with smaller ZIFs, eight). The selected data source is shown in the top right-hand corner of the display and below it the numbered range of the ZIF status currently displayed. The bottom row of hexadecimal values displayed to the left of this indicates the actual values found in the tested devices. The presence of a group of 'X's in both rows indicates that no device has been found in this ZIF socket.

```
00040 FF FF FF FF FF XX FF FF     <ROM>
      FF FF FF FF FF XX FF FF  Zifs 9-16
```

The **UP/DOWN** cursor keys may be used to reveal previous/later pages and the **LEFT/RIGHT** keys used to move the display across the array of copy ZIFs. The **MENU** key may be used to return to the 'Func' sub-menu.

```
VERIFY DONE
Hit any Key to Continue
```

Note:- The LED ZIF status indicators do not in this instance indicate the success or failure of the tested devices.

4.8.5 Erase

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **Erase** to erase devices in copy ZIFs.

This function is used to erase electrically erasable devices only. This menu option will appear only when such a device has been selected using the main menu 'Device' function (see section 4.4 'Device Selection').

ERASE ERROR Master ZIF Empty

Should this function be attempted without a device loaded into the master ZIF the above message will be displayed.

ERASE BUSY

If a device uses an auto erase function prior to writing each byte during programming and does not support chip erase the XR16 will refuse to erase the device.

4.8.6 Cyclic Redundancy Check

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **CRC** to execute the cyclic redundancy check.

This function performs a cyclic redundancy check using the chosen data source. The result is then compared with the same for the devices in the copy ZIF sockets.

4.8.6.1 Cyclic Redundancy Check using RAM

```
CYCLIC REDUNDANCY CHECK BUSY
Enter RAM Start Address _____
```

The display the prompts for the start address of the system RAM to be CRC checked.

```
CYCLIC REDUNDANCY CHECK BUSY
Enter ROM Start Address _____
```

The user is then prompted for the device ROM start address to be CRC checked.

```
CYCLIC REDUNDANCY CHECK BUSY
Block Length (default = DEVICE) _____
```

The display then prompts for the length, in bytes, of the block to CRC checked. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

```
CYCLIC REDUNDANCY CHECK BUSY
03A
```

Then the display indicates the progress of the check by use of a memory block counter.

```
CYCLIC REDUNDANCY CHECK DONE
CRC = 4132
```

The calculated CRC for the chosen block of memory is displayed on completion of the check and the system waits for a key to be pressed before returning to the 'Func' sub-menu. The LED ZIF indicators are used to display which of the copy ZIF devices have this calculated CRC result (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.6.2 Cyclic Redundancy Check using Master ZIF

```
CYCLIC REDUNDANCY CHECK BUSY
03A
```

The display indicates the progress of the check by use of a memory block counter.

```
CYCLIC REDUNDANCY CHECK ERROR
Master ZIF Empty
```

If the CRC test is attempted without first inserting the required master device in the master ZIF socket the above error will be reported. Press any key to return to the 'Funcs' sub-menu.

```
CYCLIC REDUNDANCY CHECK DONE
CRC = 4132
```

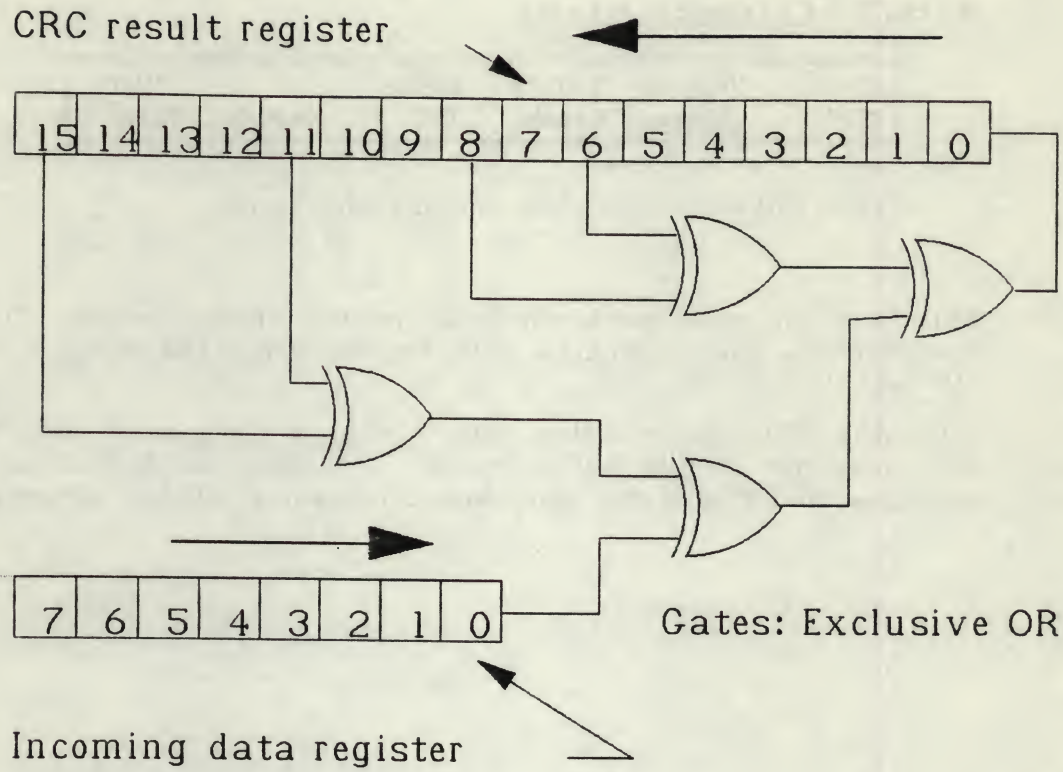
The calculated CRC is displayed on completion of the check and the system waits of a key to be pressed before returning to the 'Func' sub-menu. The LED ZIF indicators are used to display the success/failure of devices, in the copy ZIFs, to have this calculated CRC result (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.6.3 Cyclic Redundancy Check Algorithm

```
crc()
{
     unsigned int   *datptr;
     unsigned int   data, bit;
     unsigned int   crc1=0;
     setadd(romstart);
     for(datptr=start, n=length; n; n--){
          data= *datptr;
          for(bit=8; bit; bit--){
               crc1=(crc1<<1)|(1&(((((crc1^(crc1<<4))>>15)
                    ^data)^(crc1>>6))^(crc1>>8)));
               data>>=1;
          }
          incadd();
     }
}
```

The above algorithm has been written using the 'C' programming language and is supplied for information only.

The algorithm may also be expressed as a block diagram.



Block Diagram of the CRC Generator

As in the above diagram a CRC is generated as follows:

* The CRC result register is initialised to all zeros.
* The first byte of data is loaded into the incoming data register.
* The exclusive or result as shown in the diagram is clocked into the CRC result register.
* The CRC result register is shifted one place to the left.
* The incoming data register is shifted one place to the right.
* The shifting process is continued until the entire incoming data byte has been shifted out.
* The next data byte is loaded into the incoming data register.
* The process is repeated until all data bytes have been processed.
* The data remaining in the CRC result register is the CRC of the data processed.

4.8.7 Checksum

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **Chksum** to execute memory checksum.

This function performs a checksum on the chosen source. The result of the check is then compared with the same for the devices in the copy ZIF sockets.

When the XR16 in operating with a system word more than 8-bits wide this checksum facility will calculate a number of checksums. For 16-bit mode two and for 32-bit mode four checksums will be calculated.

4.8.7.1 Checksum using RAM

```
CHECKSUM BUSY
Enter RAM Start Address _____
```

After pressing 'Chksum' the display then prompts for the start address of the system RAM to be checksum tested.

```
CHECKSUM BUSY
Enter ROM Start Address _____
```

The user is then prompted for the device ROM start address to be checksum tested.

```
CHECKSUM BUSY
Block Length (default = DEVICE) _____
```

The display then prompts for the length of the block to checksum tested, in bytes. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

```
CHECKSUM BUSY
03A
```

Then the display indicates the progress of the check by use of a memory block counter.

```
CHECKSUM DONE
Chksum = 0FF000
```

The calculated checksum for the chosen block of memory is displayed on completion of the check and the system waits of a key to be pressed before returning to the 'Func' sub-menu. The LED ZIF indicators are used to display the success/failure of devices, in the copy ZIFs, to have this calculated checksum result (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.7.2 Checksum using Master ZIF

```
CHECKSUM BUSY
03A
```

After pressing 'Chksum' the display indicates the progress of the check by use of a memory block counter.

```
CHECKSUM ERROR
Master ZIF Empty
```

If the checksum test is attempted without first inserting the required master device in the master ZIF socket the above error will be reported. Press any key to return to the 'Funcs' sub-menu.

```
CHECKSUM DONE
Chksum = 401B3D
```

The calculated checksum is displayed on completion of the check and the system waits of a key to be pressed before returning to the 'Func' sub-menu. The LED ZIF indicators are used to display the success/failure of devices, in the copy ZIFs, to have this calculated CRC result (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.7.3 Checksum Algorithm

The XR16 performs a checksum which consists of the addition of the 8-bit data bytes in memory into a 24-bit result. Any carry from bit 24 is discarded.

The example below shows the addition of the data 4F 2E FF A1 A2:

```
   4F
+ 2E
+ FF
+ A1
+ A2
------
0002BF
```

Note:- Checksums may be performed over all or part of the master device, or over any section of the system RAM.

4.8.8 Blank Check

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **Blank** to execute blank device check.

This function compares each location of each of the copy devices, over the selected address range, with the known blank state for the devices. When used over the entire address range of the devices it is an effective test for complete erasure.

```
BLANK CHECK BUSY
06A
```

The function then displays that it is performing the blank device test and indicates its progress by use of a memory block counter.

```
BLANK CHECK DONE
Hit any Key to Continue
```

If 1 or more of the devices in the copy sockets has failed the blank check. The flashing LED(s) below the sockets indicate which devices have failed (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.9 Illegal Bit Check

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **IBC** to execute the illegal bit check test.

This function performs an illegal bit check between the chosen source and the copy ZIFs to determine whether the copy devices may be programmed with the desired data.

Should one or more of the devices in the copy sockets have failed the check the flashing LED(s) below the sockets indicate which devices have done so (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.9.1 Illegal Bit Check using RAM

```
ILLEGAL BIT CHECK BUSY
Enter RAM Start Address _____
```

After pressing **IBC** the display then prompts for the start address of the system RAM to be tested.

```
ILLEGAL BIT CHECK BUSY
Enter ROM Start Address _____
```

The user is then prompted for the device ROM start address to be illegal bit checked.

```
ILLEGAL BIT CHECK BUSY
Block Length (default = DEVICE) _____
```

The display then prompts for, in bytes, the length of the block to be bit tested. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

```
ILLEGAL BIT CHECK BUSY
          03A
```

Then the display indicates the progress of the check by use of a memory block counter.

```
ILLEGAL BIT CHECK DONE
Hit any Key to Continue
```

Should one or more of the devices in the copy sockets have failed the check the flashing LED(s) below the sockets indicate which devices have done so (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.9.2 Illegal Bit Check using Master ZIF

```
ILLEGAL BIT CHECK BUSY
        03A
```

After pressing **IBC** the display indicates the progress of the check by use of a memory block counter.

```
ILLEGAL BIT CHECK ERROR
Master ZIF Empty
```

If the illegal bit check is attempted without first inserting the required master device in the master ZIF socket the above error will be reported. Press any key to return to the 'Funcs' sub-menu.

```
ILLEGAL BIT CHECK DONE
Hit any Key to Continue
```

Should one or more of the devices in the copy sockets have failed the check the flashing LED(s) below the sockets indicate which devices have done so (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.10 Verify

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **Verify** to execute device program verify.

This function performs a verify of the data held in the copy ZIFs with data taken from the source data area (see section 4.8.1 'Data Source Selection').

If one or more of the devices in the copy sockets has failed the blank check. The flashing LED(s) below the sockets indicate which devices have failed (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.10.1 Verify using RAM

```
VERIFY BUSY
Enter RAM Start Address _____
```

After pressing **Verify** the display then prompts for the start address of the system RAM to be tested.

```
VERIFY BUSY
Enter ROM Start Address _____
```

The user is then prompted for the device ROM start address to be verified.

```
VERIFY BUSY
Block Length (default = DEVICE) _____
```

The display then prompts for, in bytes, the length of the block to be tested. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

```
VERIFY BUSY
03A
```

Then the display indicates the progress of the check by use of a memory block counter.

```
VERIFY DONE
Hit any Key to Continue
```

Should one or more of the devices in the copy sockets have failed the verify test the flashing LED(s) below the sockets indicate which devices have done so (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.10.2 Verify using Master ZIF

```
VERIFY BUSY
03A
```

Then the display indicates the progress of the check by use of a memory block counter.

```
VERIFY ERROR
Master Zif Empty
```

If the verify test is attempted without first inserting the required master device in the master ZIF socket the above error will be reported. Press any key to return to the 'Funcs' sub-menu.

```
VERIFY DONE
Hit any Key to Continue
```

Should one or more of the devices in the copy sockets have failed the verify test the flashing LED(s) below the sockets indicate which devices have done so (see section 5.7 'Appendix G LED ZIF Status Indicators').

4.8.11 Store

Chain	Program	VisVfy	Erase		<RAM>
CRC	Chksum	Blank	IBC	Verify	Store

Press **Store** to store master device in system RAM.

This function reads the contents of the device held in the master ZIF socket and stores it in the system RAM. A 24-bit checksum of the stored data is displayed on the LCD at the end of the function.

```
STORE BUSY
Enter RAM Start Address _____
```

The display prompts for the address at which the data on the device is to be stored.

```
STORE BUSY
Enter ROM Start Address _____
```

This is followed by a prompt for the ROM address from which the store process is to begin.

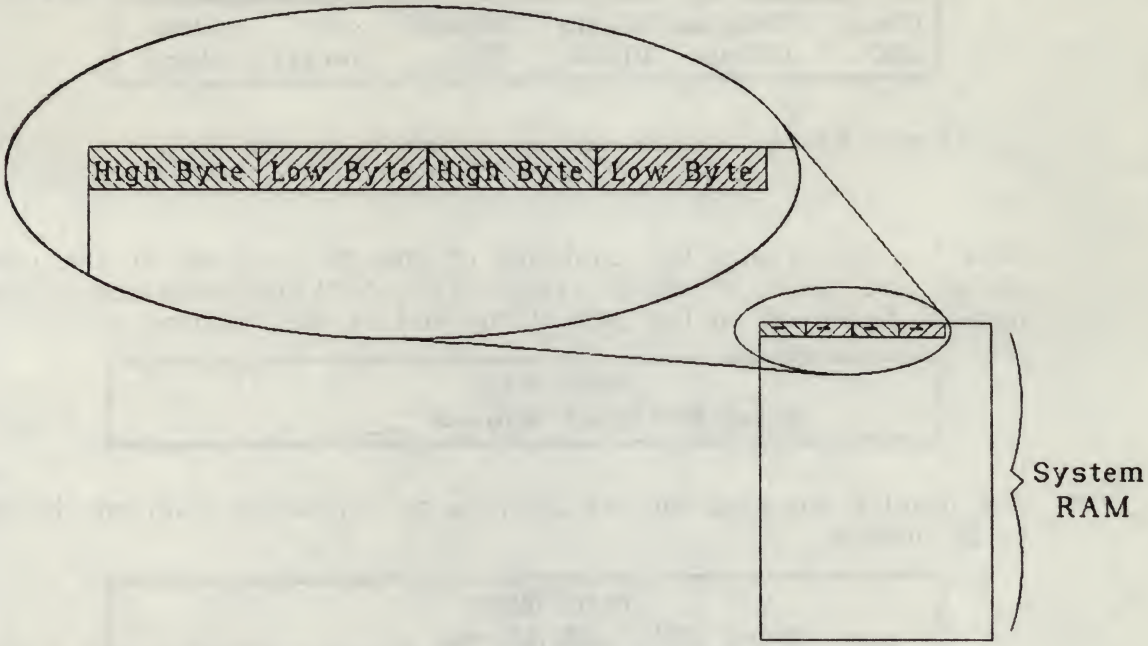
```
STORE BUSY
Block Length (default = DEVICE) _____
```

The block length, in bytes, to be copied from the device will then be requested. The system will default to the full size of the currently selected device (see section 4.4 'Device Selection') should no value be entered. This value must be entered in hexadecimal.

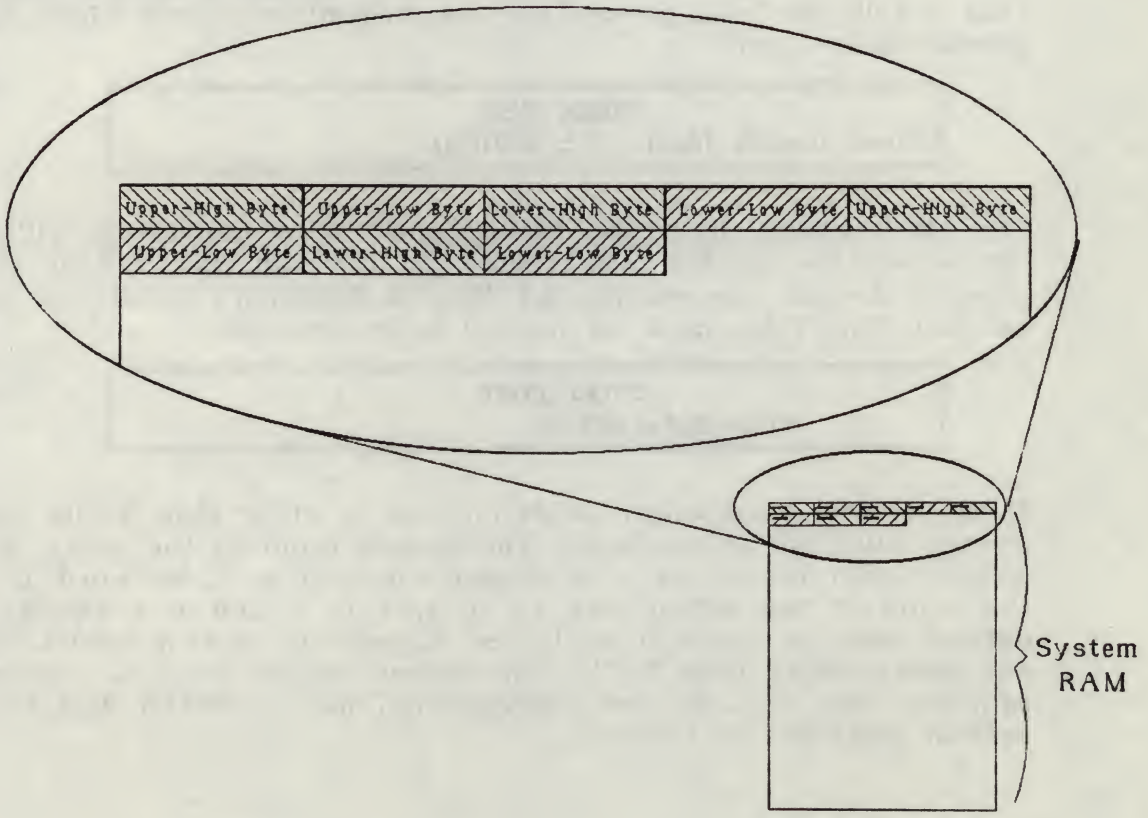
```
STORE BUSY
Enter Byte Offset _____
```

If the system word length is defined to be other than 8-bits the above prompt will also be displayed. The system requires the entry of a byte offset within the space of a system word. If a 32-bit word is defined the value of this offset may be 0, 1, 2 or 3 and if a 16-bit word is defined may be either 0 or 1 (see diagram in section 4.8.3.3 '16/32-Bit Set Programming from RAM'). This offset may be used to construct the required data map, for set programming, in the system RAM space (see section diagram that follows).

16-Bit Data Memory Map



32-Bit Data Memory Map



```
STORE BUSY
03A
```

Then the display indicates the progress of the check by use of a memory block counter.

```
STORE ERROR
Master ZIF Empty
```

If a call to store is attempted without first inserting the required master device in the master ZIF socket the above error will be reported. Press any key to return to the 'Funcs' sub-menu.

```
STORE FAIL
Hit any Key to Continue
```

Should the store attempt be unsuccessful for any reason the above message will be displayed.

```
STORE DONE
Chksum = 072239
```

When the store is completed the calculated checksum will be displayed with the completion message. Press any key to return to 'Funcs' sub-menu.

Example of Data Store:

It is required that a set of four Hitachi 27128 devices (32-bit data set) be stored in the system RAM space in preparation for set programming.

Firstly the system word length must be defined to be 32-bits wide (see section 4.5.8 'System Word Length') and secondly the device type (see section 4.4 'Device Selection'). After this definition the **Store** function may be executed.

```
STORE BUSY
Enter RAM Start Address ____0
```

As the sum of the memory areas required by the four devices is quite large RAM start seems the best place to position the data. Enter zero as the RAM start address.

```
STORE BUSY
Enter ROM Start Address ____0
```

The whole device is required to be copied and so this must also be zero.

```
STORE BUSY
Block Length (default = DEVICE) ____0
```

To down-load the whole device enter zero again to make use of the default available for this field.

```
STORE BUSY
Enter Byte Offset ____0
```

To down-load the first device (containing the upper-high byte) a byte offset of zero is required. The upper-low byte will require a byte offset of one, the lower-high byte an offset of two and the lower-low byte an offset of three (see diagram in section 4.8.3.3 '16/32-Bit Set Programming from RAM').

```
STORE BUSY
03A
```

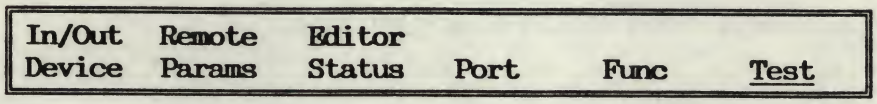
Then the display indicates the progress of the check by use of a memory block counter.

```
STORE DONE
Chksum = 072239
```

The checksum of the down-loaded device then displayed and the store sequence for that device is complete. This sequence must be repeated for all the devices in the set with care being taken to enter the correct byte offset for each device down-loaded.

4.9 System Tests

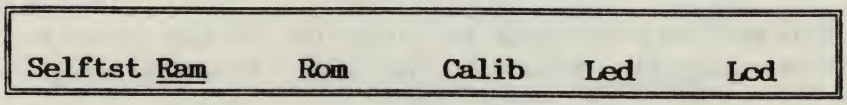
The **Test** function may be used to check up on the various portions of the system. Power supply calibration is also available as part of the test function.



Press **Test** to enter the test function sub-menu.

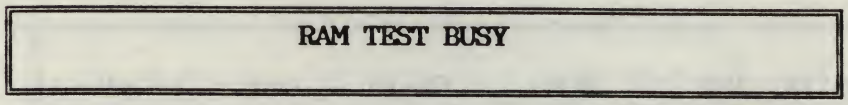
Upon selection of this function the above sub-menu will be displayed. This then enables the required system test function to be selected.

4.9.1 RAM Test



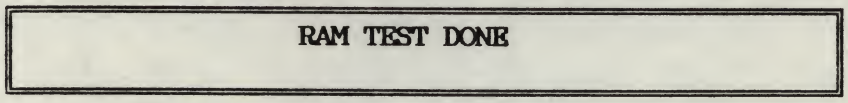
Press **Ram** to test the system RAM space.

This function tests the read/write accuracy of the whole of the system RAM.



When the RAM test function is executed the display will inform the user that the system is now busy carrying out the test.

Should the RAM test discover a memory fault the above message will be displayed and the system will wait for a key to be pressed by the user.



As the check is completed the display will change to inform the user of this and will wait for a key to be pressed before returning to the 'Test' sub-menu.

4.9.2 ROM Test

Selftst	Ram	Rom	Calib	Led	Lcd

Press **Rom** to test the system ROM space.

This function may be used to test the system ROM space for corruption of stored data.

```
ROM TEST BUSY
```

When the ROM test function is executed the display will inform the user that the system is now busy carrying out the test.

```
ROM FAULT
```

Should the ROM test discover a memory fault the above message will be displayed and the system will wait for a key to be pressed by the user.

```
ROM TEST DONE
Chksum = 074701
```

As the check is completed the display will inform the user of the verified checksum and will wait for a key to be pressed before returning to the 'Test' sub-menu.

4.9.3 Calibration

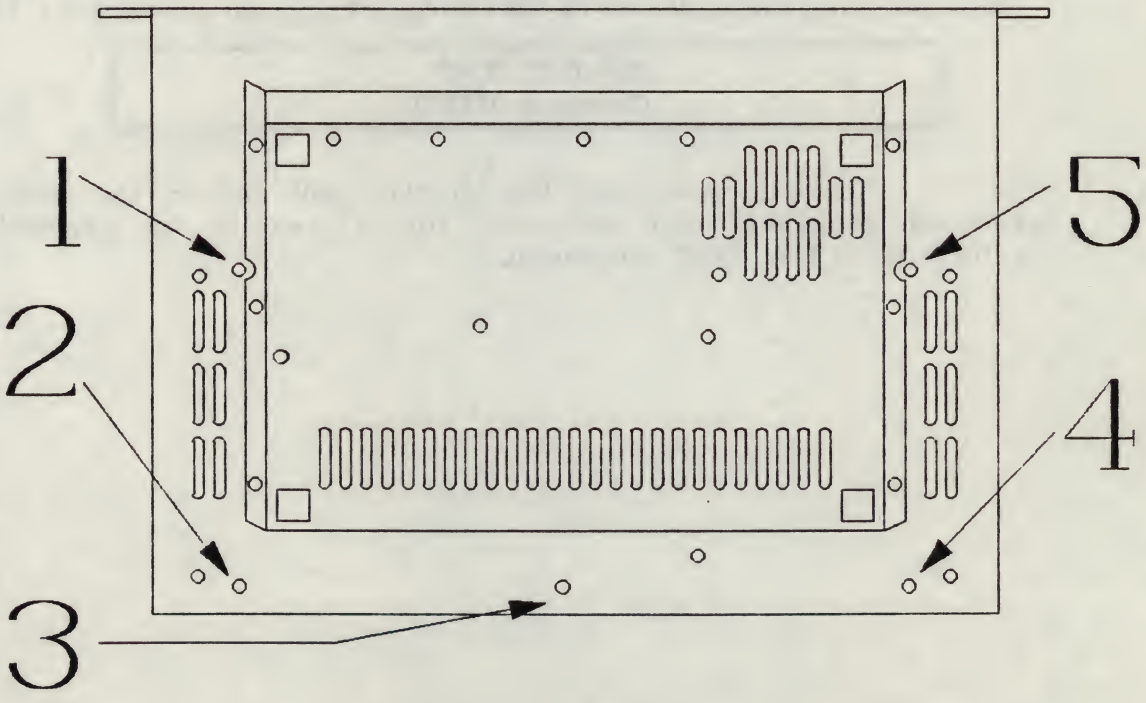
Selftst	Ram	Rom	Calib	Led	Lcd

Press **Calib** to calibrate the system power supply.

This function is used to set the internal reference voltage to 12v. All other system voltages are derived from this and so it should be the only adjustment required.

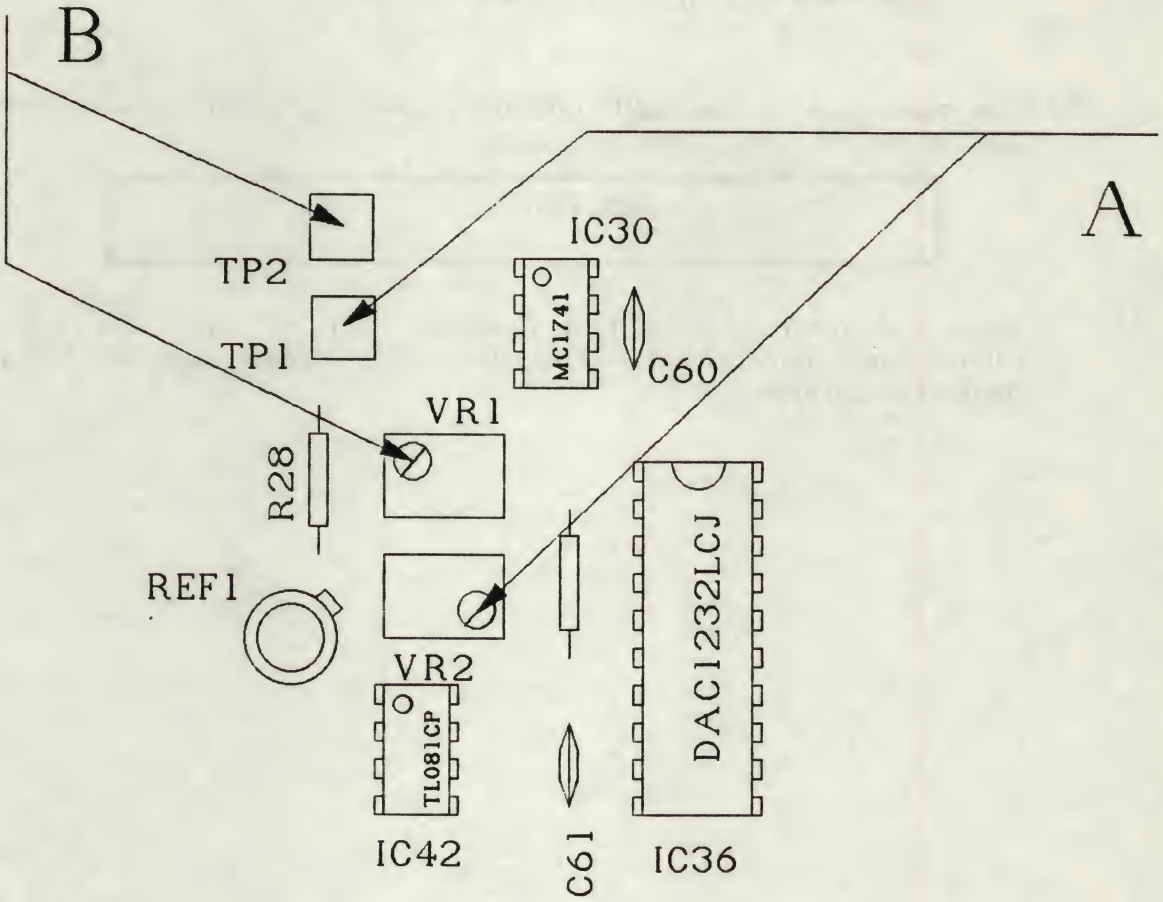
CALIBRATE POWER SUPPLY
Adjust POT to give 12v o/p, then ENTER

After selecting **Calib** the display will appear as above. The user must then detach the XR16 keyboard unit from the remainder of the case. This may be achieved by the removal of 5 screws located on the under-side of the system as shown in the diagram that follows. The system should not, as yet, be powered down.



The Under-side of the XR16

The keyboard is attached, internally, to the system PCB by a length of connector braid. Once the screws have been removed the keyboard will, after the system has been righted, lift up and off and may be lain face down whilst still connected.

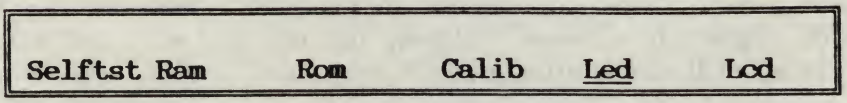


XR16 Calibration

Using the diagram above locate the 2 calibration touch pads labelled as 'TP1' and 'TP2'. Once these have been located first place your DVM probe on 'TP1' and adjust 'VR2' until your DVM reads 6.20v (location annotated as A). Then place the DVM probe on 'TP2' and adjust 'VR1' until your DVM reads 12.00v (location annotated as B). When both levels have been set press **ENTER** on the XR16 keyboard and then power down the system.

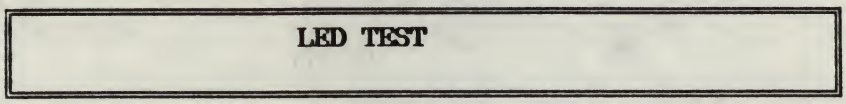
Finally replace the keyboard on the system and power up the reassembled XR16.

4.9.4 LED Test



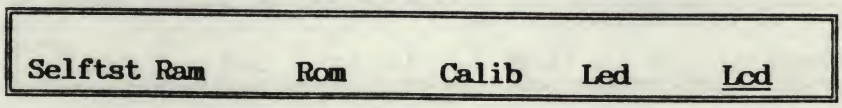
Press **Led** to test the LED ZIF indicators.

The operation of the LED indicators used by your chosen XR16 module can be tested using this function.



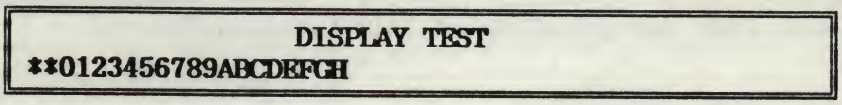
Upon execution of this test function each of the LEDs will then be turned on then off in turn. After this sweep test all LEDs will be flashed together.

4.9.5 LCD Test



Press **Lcd** to test the LCD display device.

The LCD is used as the means to display most information on the XR16 system. This test function will, when executed, enable the user ensure correct operation of this device.



The function will then run a short test sequence on the LCD display which will enable an observing operator to pin-point errors on this device.

4.9.6 Self Test

Selftst	Ram	Rom	Calib	Led	Lcd

Press **Selftst** to execute a test of the whole system.

The self test function enables the user, with a single key depression, to run a check across the whole of the XR16 system. It simply calls in turn each of the checks in the 'Test' sub-menu. Hence any previously documented system waits, in the individual test functions, will remain present during the execution of this function.

5 Appendices

5.1 Appendix A Serial Data Transfer

Serial data transfer is one process by which information is passed between pieces of electronic equipment.

As the term 'serial' implies the data is transmitted, in a stream, one piece or bit at a time. Hence in theory at least it is possible to transmit data down a single wire, making it a low cost communication technique. In order to allow different pieces of equipment to communicate an industry 'standard' has arisen: RS-232 or CCITT V.24. Unfortunately for historical reasons it is not a well defined standard, with many variations existing. The implementation on the XR16 is the result of many years experience and should allow for simple interfacing to the majority of equipment.

There are many texts on the subject of RS-232 communication which may prove to be of assistance to those new to RS-232.

Note: All information in this manual on serial interfacing relates equally to RS-232C and CCITT V.24, for the sake of simplicity the term RS-232 will be used throughout the manual.

5.1.1 Setting up the RS-232C Communications Link

In order to establish communication between the XR16 and your equipment via the RS-232 link you will need to obtain the following information:

* Determine configuration of cable. To do this you will need to refer to the manual for the equipment which you are connecting to the XR16, for XR16 pinout details and the description of signals (see section 5.1.2 'Signal Definition and Pinout').
* Determine the Data Transfer Format which you will be using. This is usually determined by the software you are using to generate the code which you wish to transfer to the XR16. The formats which the XR16 will accept are documented in section 5.2 'Appendix B Data Transfer Formats'.
* Determine the word format that you will be using. This may be determined by other equipment which you have attached to your system or by in-house standards. If you are free to determine your own word format it is worth bearing in mind that the less bits there are in a word the faster the transmission will proceed (see section 5.1.4 'Word Format').
* Determine the Parity settings which you will be using. This again will depend on your system. It is fairly common when using short cable lengths to neglect parity, but if you are using an unchecked binary format you should consider that parity is the only form of error checking available to you.
* Determine the baud rate at which your system sends data. The most commonly used baud rate is 9600. The XR16 will operate at baud rates of up to 19k2.

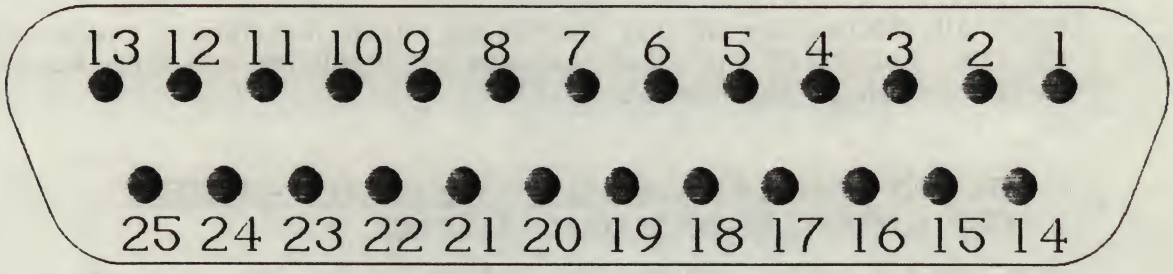
* Determine which type of handshaking your system uses. This is often a function of operating systems, so you will need to study your system documentation to decide which type your system uses. The XR16 handshaking options are documented in section 5.1.6 'Flow Control'.

When you have this information available you can configure the serial port as described in section 4.7 'Port Protocol'.

5.1.2 Signal Definitions and Pinout

The XR16 has a serial port which is mounted on the front of the machine (see section 2.1 'View of XR16'). The port is a 25-pin female Dee type connector.

Serial Port Pinout



Pin	Direction	Signal Name	Signal Description
1	N/A	Case Ground	Protective ground
2	OUT	TXD	Transmit Data
3	IN	RXD	Receive Data
4	OUT	RTS	Request to Send
5	IN	CTS	Clear to Send
6	IN	DSR	Data Set Ready
7	N/A	Signal Ground	Return Ground for Signals
8	PULL UP	DCD	Data Carrier Detect
20	OUT	DTR	Data Terminal Ready

Note:- All input handshake are internally pulled up. This means that these lines will provide a ready input to the XR16 if they are not connected.

Signal Definitions

The XR16 transmits data via 'pin 2' (TXD) and receives data via 'pin 3' (RXD).

When hardware handshaking has been enabled control signals are output from the XR16 via 'pin 4' (RTS) and 'pin 20' (DTR). Handshake control signals transmitted by the receiving equipment are received, by the XR16, via 'pin 5' (CTS) and 'pin 6' (DSR).

The flow of data from the XR16 is controlled by 'pin 5' (CTS) and 'pin 6' (DSR) if hardware handshaking is enabled. When either one of these is pulled low (less than 3v), the XR16 will finish transmitting the current data byte and then halt data transmission until both pins are again high (greater than 3v).

The flow of data into the XR16 should be controlled by 'pin 4' (RTS) and 'pin 20' (DTR) when hardware handshaking has been enabled. The transmitting equipment must therefore make use of one or both of these signals if such handshaking is to be implemented.

5.1.3 Signal Levels

Input Signal Levels

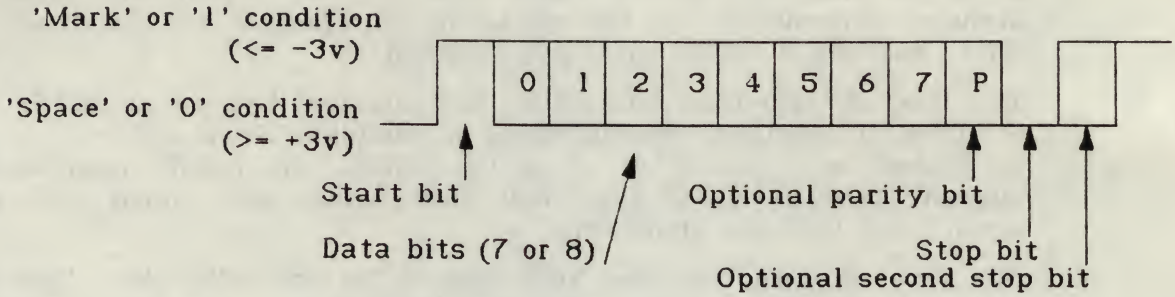
Signal Level	Max (V)	Min (V)
Mark or 1 condition	-3	-15
Space or 0 condition	+15	+3

Output Signal Levels

Signal Level1	Nominal (V)
Mark or 1 condition	-12
Space or 0 condition	+12

5.1.4 Word Format

Data transmitted via an asynchronous serial link must be packaged with additional information to maintain synchronisation between the sending and receiving machines.



Note:- Word format should be defined to suit that of the equipment with which you wish to communicate (see section 4.7 'Port Protocol').

5.1.5 Parity

In each data word there may be an optional last bit, prior to the stop bit; the parity bit. The purpose of the parity bit is to allow for checking of the accuracy with which the data was sent between the two machines.

Parity may be either odd or even. The parity bit is added to the other bits in the data word (modulo 2), if the result is a 1 then the parity is odd, a zero and the parity is even.

It is strongly recommended that if operating in a noisy environment or over very long lines that you use the parity facility.

5.1.6 Flow Control

The XR16 implements both hardware and software handshaking.

5.1.6.1 Hardware Handshaking

Handshaking whilst receiving data:
When the XR16 is ready to receive data it places both of its output handshake lines (see section 5.1.2 'Signal Definition and Pinout) in the space condition (+12V). If at any time the XR16 cannot keep up with the data rate it will change both output handshake lines to the mark condition (-12V). Thus in order to correctly send data to the XR16 you should only transmit data when both handshaking lines are in the space condition (+12V).

Handshaking whilst transmitting data:
The XR16 will only transmit data when both of its handshake input lines (see section 5.1.2 'Signal Definitions and Pinout') are in the space condition (+12V).

5.1.6.2 Software Handshaking.

Handshaking whilst receiving:

When the XR16 is ready to receive data it sends the XON character (11H). As the input buffer becomes full the XR16 transmits XOFF (13H) characters until the data input stops. When there is space in the buffer the XR16 sends the XON character again.

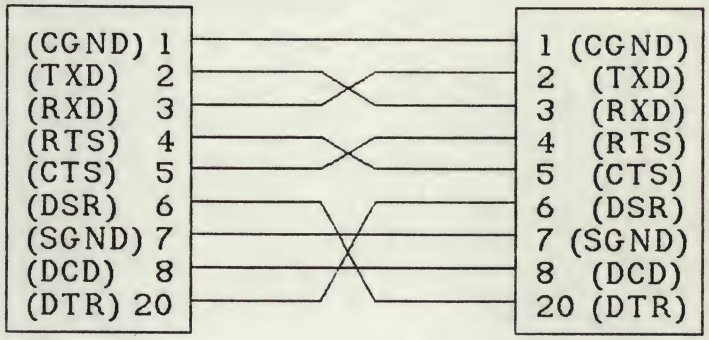
Handshaking whilst transmitting:

The XR16 transmits until it receives an XOFF character. It then suspends transmission until an XON character is received.

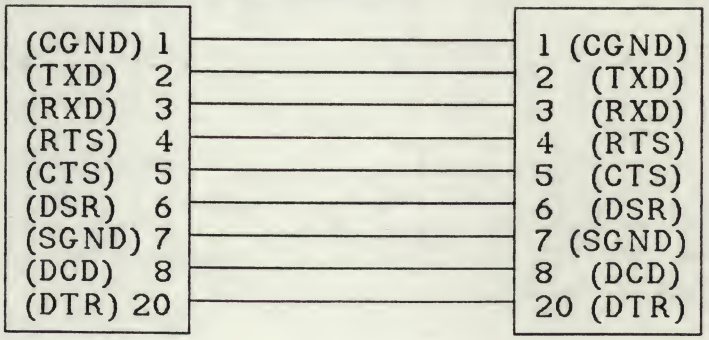
5.1.7 Cable Configurations

In the field of serial communication many different cable configurations have been implemented. Some of the more common of these are illustrated below.

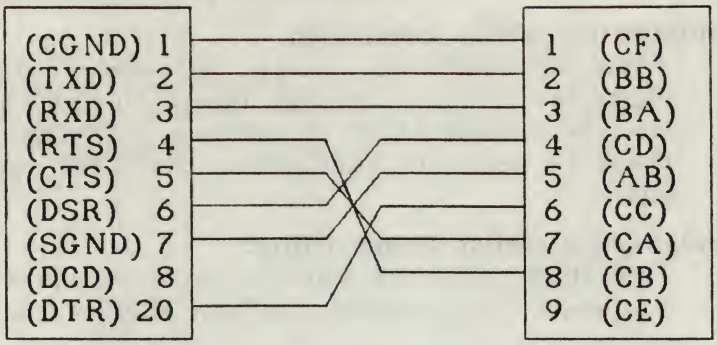
XR16-DTE (25-Pin D-Type) Configuration



XR16-DCE (25-Pin D-Type) Configuration



XR16-IBM (9-Pin D-Type) Configuration



5.2 Appendix B Data Transfer Formats

Data transfer formats may be either an ASCII format or a binary format. In the case of both ASCII and binary format data in transmitted as hexadecimal words (either 7 or 8-bits long for ASCII formats and always 8-bits for binary formats).

When using an ASCII format, two hexadecimal words are required per byte of data transmitted (each byte being the ASCII value of one nibble of data). For binary formats a single byte is transmitted for each data byte. Both types of format also insert various record markers and/or counters (depending on the format used) as a data stream is transmitted as an aid to error detection.

5.2.1 Intel Hex Data Format

The Intel Hex format is a widely used format for the transfer of binary data. It transmits the data as short data records in ASCII code each, record having a checksum in order to ensure integrity of the data.

There are several record types within the definition of Intel Hex, but the XR16 only uses three of them. These are: type 00 -data record, type 01 the end of file record and type 02 the extended address record. If the XR16 receives any other records it just discards them.

5.2.1.1 Intel Data Record Format (type 00)

Byte	Description
1	Colon (:) delimiter
2 - 3	Number of binary bytes of data in this record. The maximum is 32 binary bytes (64 ASCII bytes).
4 - 5	Most significant byte of the start address of the data.
6 - 7	Least significant byte of the start address of the data.
8 - 9	ASCII zeros. The "record type" for a data record.
10 -	Data bytes. Each binary byte is sent as two ASCII characters each one representing a nibble of the hexadecimal representation of the byte.
Last two bytes	Checksum of all bytes in the record, excluding the delimiter, carriage return and line feed. The checksum is the negative of the modulo 256 binary sum of all of the bytes in the record. CR,LF Carriage return, line feed.

5.2.1.2 Intel Extended address record (Type 02)

Byte	Description
1	Colon (:) delimiter
2 - 3	"02" The record length
4 - 7	Ascii zeros
8 - 9	Record type "02"
10 - 13	USBA Upper segment base address (The top 16 bits of a 20 bit address) It is used in Intel's 16 bit data records. If no 02 records are sent the USBA is set to zero. If a USBA is specified then the bottom 12 bits are added to the offset address of the data records.
14 - 15	Checksum of all bytes in the record, excluding the delimiter, carriage return and line feed. The checksum is the negative of the modulo 256 binary sum of all of the bytes in the record.
16 - 17	CR,LF Carriage return, line feed.

5.2.1.3 Intel End of File Record (Type 01)

Byte	Description
1	Colon (:) delimiter
2 - 3	ASCII zeros.
4 - 5	Most significant byte of transfer address (not used by XR16 ; set to zeros).
6 - 7	Least significant byte of transfer address (not used by XR16 ; set to zeros).
8 - 9	Record type 01. Indicates end of record
10 - 11	Checksum.
12 - 13	CR,LF Carriage return, line feed.

5.2.1.4 Example of Intel Hex

Given the data "**23 45 AF B1 D0 77**" to be sent as an Intel Hex Record to start at address 0000. The Record would be:

:060000002345AFB1D077EB<CR><LF>

Which may be broken down as:

Delimiter	:
Number of Bytes in the Record	06
Start Address High	00
Start Address Low	00
Record Type	00
Data	23
	45
	AF
	B1
	D0
	77
Checksum	EB
Carriage Return	CR
Line Feed	LF

Where the Checksum is calculated as follows:

CS = 06+00+00+00+23+45+AF+B1+D0+77	= 315
Modulo 256	= 15
Negative	= EB

NB: The above checksum calculation was performed in hexadecimal.

5.2.1.5 Upper Segment Base Addresses (USBA)

The Intel hexadecimal records which may be received by the XR16 may be either the standard 8-bit format (record types 00 & 01) or the extended 16-bit format (additional record type 02).

The USBA is a 16-bit number which is used to set the current segment base. (This terminology is derived from the Intel 8086). In effect this means that the 16-bit number is shifted left four times and added to the 16-bit address of the type 00 data records. This results in a 20-bit address.

e.g.

USBA =	1263H
Left shifted 4 bits =	12630H
Address in data record =	3334H
Actual address of data (USBA + Record address) =	15964H

5.2.2 Motorola Exorciser or "S" Format

The Motorola "S" format provides for the transmission of data in printable ASCII format. The data is divided into records. The XR16 recognises and uses three types of record, these are: "S1" and "S2" the data records, and "S8" and "S9" the end of file records.

5.2.2.1 Exorciser Data Record (type S1)

Byte	Description
1	"S" character delimiter.
2	ASCII 1. The record type for data.
3 - 4	Byte count. The number of binary data bytes in the record plus three (1 for checksum and 2 for address).
5 - 6	Most significant byte of the start address of the data record.
7 - 8	Least significant byte of the start address of the data record.
9 -	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the hexadecimal representation of the byte.
Last two bytes	Checksum of all bytes in the record excluding the delimiter and record type. The checksum is the 1's complement (NOT) of the modulo 256 binary sum of the bytes in the record.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.2.2 Exorciser Data Record (type S2)

Byte	Description
1	"S" character delimiter
2	ASCII 2. The record type for data.
3 - 4	Byte Count. The number of binary bytes in the record plus four (1 for checksum and 3 for address).
5 - 6	Most significant byte of start address of the data record
7 - 8	Next most significant byte of start address of the data record
9 - 10	Least significant byte of start address of the data record
11 -	Data bytes. Each byte is set as two ASCII characters, each representing one nibble of the hexadecimal representation of the byte.
Last two bytes	Checksum of all bytes in the record excluding the delimiter and record type. The checksum is the 1's complement (NOT) of the modulo 256 binary sum of the bytes in the record.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.2.3 Exorciser End of File Record (type S8)

Byte	Description
1	"S" delimiter
2	ASCII 8 Indicates end of file record
3 - 4	Byte count = 04 in end of file record
5 - 6	Most significant byte of start address (not used in the XR16 ; set to zero)
7 - 8	Next most significant byte of start address (not used in the XR16 ; set to zero)
9 - 10	Least significant byte of start address (not used by the XR16 ; set to zero).
11 - 12	Checksum
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.2.4 Exorciser End of File Record (type S9)

Byte	Description
1	"S" delimiter
2	ASCII 9 Indicates end of file record
3 - 4	Byte count = 03 in end of file record
5 - 6	Most significant byte of start address (not used in the XR16 ; set to zero)
7 - 8	Least significant byte of start address (not used by the XR16 ; set to zero).
9 - 10	Checksum
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.2.5 Example of Motorola Exorciser Format

A Motorola record consisting of the data **67 A0 4A 2B** to start at 213F would be:

S107213F67A04A2B1C<CR><LF>

Which consists of:

Delimiter	S
Record Type	1
Byte Count (Data + 3)	07
Start Address High	21
Start Address Low	3F
Data	67
	A0
	4A
	2B
Checksum	1C
Carriage Return	CR
Line Feed	LF

Where the Checksum is calculated as follows:

CS = 07+21+3F+67+A0+4A+2B =	1E3
Modulo 256 =	E3
1's Complement =	1C

NB: The above calculations were performed in hexadecimal

5.2.3 GP Binary Format

This is a simple format devised by GP specifically for users writing there own formats. It is designed to be as simple as reasonably possible. All data is sent in 8-bit binary, LSB first.

5.2.3.1 GP binary Record

Byte	Description
1	Least significant byte of the block length.
2	Most significant byte of block length.
3	Least significant byte of the checksum.
4	Most significant byte of the checksum.
5 -	Data bytes.

The block length is the number of bytes in the data record.

The checksum is the modulo 65536 binary sum of the data being transferred.

5.2.3.2 Example of GP Binary Data Format

A GP Binary record to send the following data **23 67 8F 2A** would be:

Low Block Length	04
High Block Length	00
Low part of Checksum	43
High part of Checksum	01
Data	23
	67
	8F
	2A

Where the Checksum was calculated as follows:
CS = 23+67+8F+2A = 143

NB: The above calculation was performed in hexadecimal

5.2.4 List Format

This format is an output only format designed primarily to drive a serial printer. Data is output as ASCII characters in rows of 16 characters, each row being preceded by the address of the first character in the row. Each row is terminated by carriage return and line feed. The data is sent in blocks of 256 bytes. After every third block a form feed is sent to prevent data being printed on the perforations of the paper.

5.2.4.1 Example of List Output Format

```
0000 E4 AA CD 00 99 C9 E5 F5 E1 F1 4F 7D ED CF 21 01
0010 21 FF FF 0A E4 C4 01 C9 22 FD 22 E4 14 C3 FF FF
```

5.2.5 Tektronix Standard Hex Format

This format provides for the transfer of data blocked into records of printable ASCII characters. There are two types of record used and recognised by the XR16.

5.2.5.1 Tektronix Standard Data Record

Byte	Description
1	"/" character; delimiter.
2 - 3	Most significant byte of the start address of the data record.
4 - 5	Least significant byte of the start address of the data record.
6 - 7	Byte count. The number of binary data bytes in the record.
8 - 9	First checksum, sum of all nibbles, modulo 256 of the six hexadecimal bytes of the load address and byte count.
10 -	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the hexadecimal representation of the byte.
Last two bytes	Checksum of all of the data bytes in the record, calculated as the modulo 256 sum of all the nibbles making up the data bytes.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.5.2 Tektronix Standard End of File Record

Byte	Description
1	"/" character; delimiter.
2 - 3	Most significant byte of start address (not used in the XR16 ; set to zero)
4 - 5	Least significant byte of start address (not used by the XR16 ; set to zero).
6 - 7	Byte count = 00 in end of file record
9 - 10	Checksum of all bytes in the record excluding the delimiter and record type. The checksum is the modulo 256 binary sum of the NIBBLES making up the bytes in the record.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.5.3 Example of Tektronix Standard Format

To send the data 23,00,A8,A9,17,04 the data format would look like:

/000006062300A8A9170436<CR><LF>

Which consists of:

Delimiter	/
Start Address	0000
Byte Count	06
Checksum of Address field	06
Data	23
	00
	A8
	A9
	17
	04
Checksum	36

Where the checksums were calculated as:

Checksum of Address = 0+0+0+0+6 = 6
Checksum of data = 2+3+0+0+A+8+A+9+1+7+0+4 = 36H

5.2.6 Tektronix Extended Hex Format

Tektronix Extended provides for the transmission of larger bodies (greater than 64K) of data in printable ASCII format. The XR16 recognises and uses two types of data record of this format, the data record and the end of file record.

5.2.6.1 Tektronix Extended Data Record

Byte	Description
1	"%" character; delimiter.
2 - 3	Number of characters in record, not including the "%" delimiter.
4	Record type character, "6" for data records.
5 - 6	Checksum calculated as the modulo 256 of the sum of all bytes making up the data record, not including the "%" delimiter or any part of the variable length load offset.
7 -	Load offset for the first image byte of the record. The length of this field may be from two to nine characters, where the first character specifies the number of character that follow. The offset is written most significant byte first.
-	Data bytes in increasing order of address.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.6.2 Tektronix Extended End of File Record

Byte	Description
1	"%" character; delimiter.
2 - 3	Number of characters in record, not including the "%" delimiter.
4	Record type character, "8" for end of file records.
5 - 6	Checksum calculated as the modulo 256 of the sum of all bytes making up the data record, not including the "%" delimiter or any part of the variable length load offset.
7 -	The start address in the same format as the load offset in the data record. The start address used is the first load offset output for the file.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.6.3 Example of Tektronix Extended Format

To send data 86,AF,E5,64,98,99,99,00 the record would look like:

%1E68E80000000086AFE56498999900<CR><LF>

Which consists of:

Delimiter	%
Record Length	1E
Record Type	6
Checksum	8E
Load Offset	8 0000 0000
Data	86 AF E5 64 98 99 99 00

Where the checksum was calculated as:

Checksum = 1+E+6+8+6+A+F+E+5+6+4+9+8+9+9+9+9+0+0 = 8E

Modulo 256 = 8E

NB: This calculation was performed in hexadecimal.

5.2.7 MOS Technology Data Format

In this format the data is divided into records and sent as printable ASCII characters. There are two types of record used and recognised by the XR16. These are the data record and the end of file record.

5.2.7.1 MOS Data Record

Byte	Description
1	";" character; delimiter
2 - 3	Byte count. The number of binary data bytes in the record.
4 - 5	Most significant byte of the start address of the data record.
6 - 7	Least significant byte of the start address of the data record.
8 -	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the hexadecimal representation of the byte.
Last four bytes	Checksum ,sum of all data bytes in the record. The checksum is the modulo 65536 binary sum of all the bytes in the record including the block length and address, but excluding the delimiter and the checksum itself. It is transmitted high byte then low byte.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.7.2 MOS End of File Record

Byte	Description
1	";" delimiter
2 - 3	Byte count = 00 in end of file record
4 - 5	Most significant byte of sum of total bytes sent in all records
6 - 7	Least significant byte of sum of total bytes sent in all records
8 - 9	Most significant byte of checksum
10 - 11	Least significant byte of the checksum of all bytes in the record excluding the delimiter and record type. The checksum is the modulo 65536 binary sum of the bytes in the record.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.7.3 Example MOS TECHNOLOGY Data Record

To send the data record 86 AF E5 64 98 99 99 00 the MOS record would be:

;08000086AFE564989999000448<CR><LF>

Which consists of:

Delimiter	;
Byte Count	08
Start Address	0000
Data	86
	AF
	E5
	64
	98
	99
	99
	00
Checksum	0450

Where the checksum is calculate as:

Checksum = 08+00+00+86+AF+E5+64+98+99+99+00 = 0450

5.2.8 Signetics Absolute Data Transfer Format

In this format data is divided into records of printable ASCII characters. The XR16 uses and recognises two types of data record. The data record and the end of file record.

5.2.8.1 Signetics Absolute Data Record

Byte	Description
1	":" character; delimiter
2 - 3	Most significant byte of the start address of the data record.
4 - 5	Least significant byte of the start address of the data record.
6 - 7	Byte count. The number of binary data bytes in the record.
8 - 9	Checksum of all the bytes in the address and data fields, calculated by EXORing each byte with the previous byte, then rotating the resultant byte left one bit.
10 -	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the hexadecimal representation of the byte.
Last two bytes	Checksum ,sum of all data bytes in the record. The checksum is calculated in the same way as the first checksum.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.8.2 Signetics Absolute End of File Record

Byte	Description
1	":" delimiter
2 - 3	Most significant byte of start address (not used in the XR16 ; set to zero)
4 - 5	Least significant byte of start address (not used by the XR16 ; set to zero).
6 - 7	Byte count = 00 in end of file record
8 - 9	Checksum of all the bytes in the address and data fields, calculated by EXORing each byte with the previous byte, then rotating the resultant byte left one bit.
CR,LF	Carriage return and line feed are output from the XR16, but are not checked when input.

5.2.8.3 Example of Signetics Absolute Data Format

To send the data 23 EE F1 2A D4 55 99 the data record would be as follows:

:0000070E23EEF12AD4559946<CR><LF>

Which consists of:

Delimiter	:
Start Address	0000
Byte Count	07
First Checksum	0E
Data	23 EE F1 2A D4 55 99
Second Checksum	46

Where the checksums are calculated as follows:

First Checksum:
(((00 EXOR 00)*2 EXOR 00)*2 EXOR 07)*2 = 0E

Second Checksum:
((((((23 EXOR EE)*2 EXOR F1)*2 EXOR 2A)*2 EXOR D4)*2 EXOR 55)*2 EXOR 99)*2 = 46

5.2.9 The ASCII Space, Comma, Apostrophe and Percent Formats

Data in these formats is transmitted in sequential, two character groups representing hexadecimal bytes followed by the execute code space, percent, apostrophe or comma. Data may be transmitted as either 4 or 8-bits. The XR16 assumes that the two characters prior to the execute code were valid ASCII hexadecimal bytes. If only one character was received prior to the execute code then a leading zero is assumed.

When the XR16 is receiving in these formats it recognises 3 types of information; these are Address information, Data and Checksum.

General: The data transmission must be preceded with an <STX> character (02H) which may then be followed immediately with data or by an address field. The transmission must be terminated with an <ETX> (03H) followed by either a checksum field or at least 16 nulls.

5.2.9.1 Data field

Each time an execute code is received the two previous bytes are assumed to be valid data. If there have not been two valid ASCII hexadecimal bytes prior to the execute code then the programmer assumes leading zeros.

5.2.9.2 Address field

When the XR16 receives a "$" followed by an "A" it then expects four ASCII hexadecimal digits giving the address of the first data field. This address must be terminated by a comma (except in the "Comma" format where it is terminated by a full stop).The input data will then be loaded, starting at this address.

5.2.9.3 Checksum field

The data field must be terminated with an <ETX> this may optionally be followed with a checksum. The checksum is expected as "$" followed by "S" followed by the four bytes of the checksum. The checksum must be terminated with a comma (or for the comma format a full stop). The checksum is calculated as the modulo 65536 sum of all of the data sent since the previous <STX>. If the checksum is not sent then at least 16 characters must follow the <STX> to prevent a time-out error.

5.2.9.4 Example of ASCII SPACE Data Transmission

<STX>$A0000,<CR><LF>
31 FF 77 C3 FF FE 76.....<ETX><CR><LF>
$S1234,<CR><LF>

5.2.9.5 Example of ASCII COMMA Data Transmission

<STX>$A0000.<CR><LF>
31,FF,77,C3,FF,FE,76.....<ETX><CR><LF>
$S1234.<CR><LF>

5.2.9.6 Example of ASCII PERCENT Data Transmission

<STX>$A0000,<CR><LF>
31%FF%77%C3%FF%FE%76.....<ETX><CR><LF>
$S1234,<CR><LF>

5.2.9.7 Example of ASCII APOSTROPHE Data Transmission

<STX>$A0000,<CR><LF>
31'FF'77'C3'FF'FE'76.....<ETX><CR><LF>
$S1234,<CR><LF>

5.2.10 BPNF,BHLF,B10F Formats

In these formats each byte of data is transmitted as an ASCII "B" followed by 8 ASCII bytes representing the bits of the data byte. Zeroes and ones are represented respectively in the two formats by: "N", "P" or "L", "H", or "0", "1". Each byte is terminated with the ASCII character "F". The data is transmitted least significant bit first. The entire data stream must be started with a non-printable <STX> and ended with a non-printable <ETX>. The data output from the XR16 is formatted to suit a list device by inserting a space between each byte, and a <CR><LF> at the end of each line of six bytes.

5.2.10.1 Example of BPNF Format

The data stream 0F,84,73,21 would be sent as:

<STX>BPPPPNNNNF BNNPNNNNPF BPPNNPPPNF BPNNNNPNNF<ETX>

5.2.10.2 Example of BHLF Format

The data stream 0F,84,73,21 would be sent as:

<STX>BHHHHLLLLF BLLHLLLLHF BHHLLHHHLF BHLLLLHLLF<ETX>

5.2.10.3 Example of B10F Format

The data stream 0F,84,73,21 would be sent as:

<STX>B11110000F B00100001F B11001110F B10000100F<ETX>

5.2.11 DEC Binary and Binary formats

In both of these formats data is transmitted as a string of binary information. The only difference in the two formats is the start of record. For Binary the record starts with any number of nulls followed by a rubout (FFH). In DEC binary the format starts with any number of rubouts followed by a null. The data after the record start is a string of binary data with no checksums, no byte counts and no print formatting. As there is no end of file delimiter the receiving machine must have been told how many bytes to expect. In the XR16 this is entered from the keyboard.

5.3 Appendix C Parallel Data Transfer

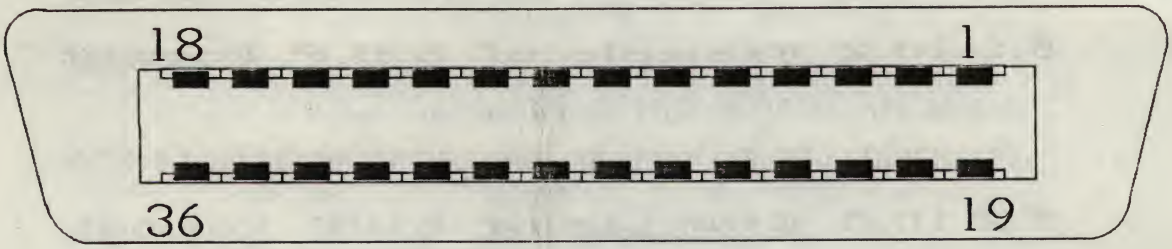
5.3.1 The Printer Interface

The XR16 printer interface is a bidirectional 'Centronics' parallel interface. This is a widely used standard, for parallel communication, and as such facilitates the interfacing of a wide range of equipment with the XR16.

5.3.2 Signal Definitions and Pinout

The printer port is the 26-pin IDC connector on the front of the XR16 (see section 2.1 'View of XR16'). It may be connected to a parallel type printer via an IDC/Parallel cable.

Parallel Port Pinout



PIN	SIGNAL	PIN	SIGNAL
1	$\overline{\text{STROBE}}$	19	TWISTED PAIR GROUND (PIN 1)
2	DATA 1	20	TWISTED PAIR GROUND (PIN 2)
3	DATA 2	21	TWISTED PAIR GROUND (PIN 3)
4	DATA 3	22	TWISTED PAIR GROUND (PIN 4)
5	DATA 4	23	TWISTED PAIR GROUND (PIN 5)
6	DATA 5	24	TWISTED PAIR GROUND (PIN 6)
7	DATA 6	25	TWISTED PAIR GROUND (PIN 7)
8	DATA 7	26	TWISTED PAIR GROUND (PIN 8)
9	DATA 8	27	TWISTED PAIR GROUND (PIN 9)
10	$\overline{\text{ACKNOWLEDGE}}$	28	TWISTED PAIR GROUND (PIN 10)
11	BUSY	29	TWISTED PAIR GROUND (PIN 11)
12	NOT CONNECTED	30	TWISTED PAIR GROUND (PIN 12)
13 - 18	NOT CONNECTED	31 - 36	NOT CONNECTED

Signal Definitions

Data is transmitted and received via the bidirectional parallel data bus ('pins 1-8').

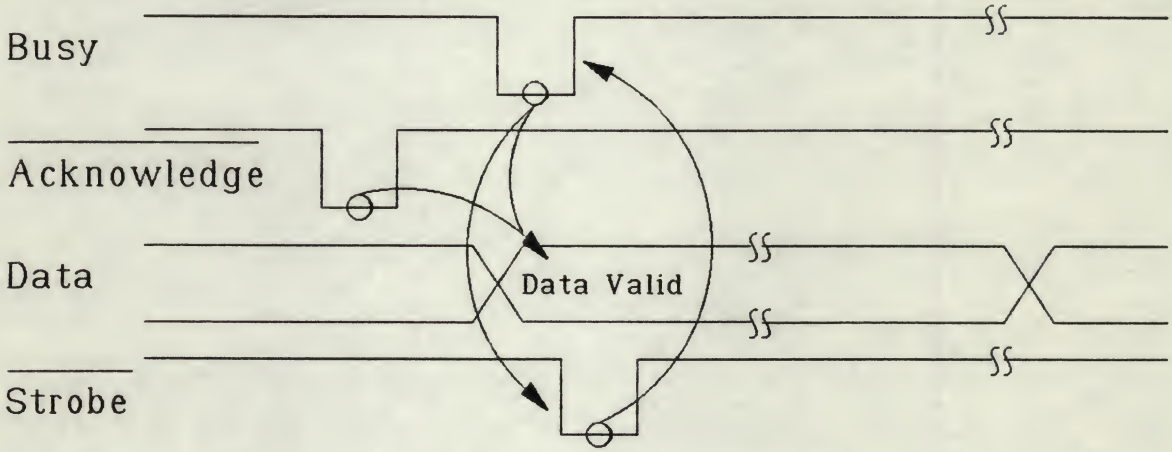
When transmitting, the XR16 will pulse '$\overline{\text{Strobe}}$' high-low-high after data is loaded onto on the data bus. When receiving, data is read following the falling (-ve) edge of the '$\overline{\text{Strobe}}$' signal.

During data transmission the 'Busy' signal is tested before each byte is actually transmitted. The '$\overline{\text{Strobe}}$' signal will not be pulsed until the 'Busy' signal is reset (low). When receiving the XR16 will set (make high) 'Busy' until it is ready to receive further data bytes.

The '$\overline{\text{Acknowledge}}$' signal is used to acknowledge the successful reception of data. This signal is taken low when a byte of data has been correctly received. When receiving data the XR16 will wait for an '$\overline{\text{Acknowledge}}$' signal before pulsing '$\overline{\text{Strobe}}$' and continuing data transmission.

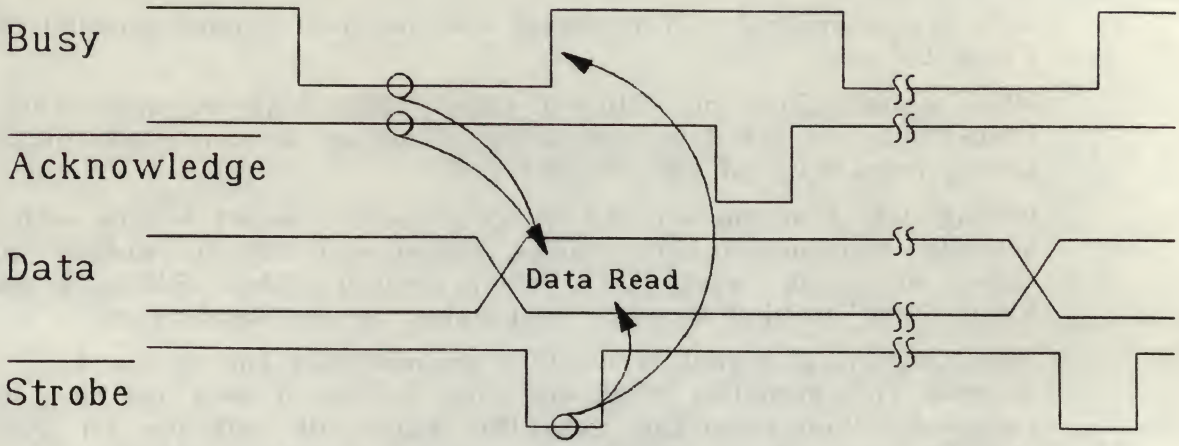
5.3.3 Timing Diagrams

Output Timing Diagram



Note:- On systems which do not use the '$\overline{\text{Acknowledge}}$' signal the 'Busy' line is used to indicate ready to receive. Also '$\overline{\text{Strobe}}$' is an active low pulse (0.5μs minimum length) and should be used to latch data into the receiver.

Input Timing Diagram



Note:- Data is read into the XR16 on the active low '$\overline{\text{Strobe}}$' signal and is acknowledged by a pulse on '$\overline{\text{Acknowledge}}$' (0.5$\mu s$ minimum length). Data should therefore be valid between 'Busy' high and '$\overline{\text{Acknowledge}}$' going low.

5.4 Appendix D Error Messages

Whenever possible error messages have been included in the text relating to the function in which they occur. However many error messages may appear at one of a number of locations and in some instances the number of possible messages made this approach impractical.

Here follows a complete list of the error messages to be found on the XR16 system:

Block cannot be copied onto itself - During the execution of the 'Editor Copy' function (see section 4.3.3 'Memory Copy') the source and the destination addresses equate to the same value.

Chain Program in 8 bit mode only - Use of the chain programming facility was attempted whilst the system word was defined other than 8-bits wide.

Checksum Mismatch - whilst receiving data the checksum calculated during reception differed from that calculated and stored in the record when the data was written (see section 5.2 'Appendix B Data Transfer Formats').

Copy ZIF Chip Enable HIGH/LOW - Voltage level applied lies outside the required tolerance (the direction of the error being indicated by HIGH/LOW).

Copy Zifs Empty - No devices were found to be present in the copy ZIFs. If devices are in place the error may be caused by an inappropriate device selection or by dirty ZIF sockets (see sections 4.4 'Device Selection' and 5.8 'Appendix H Cleaning ZIFs').

Copy ZIF High Data Bus <byte> - Inappropriate levels have been found present on the lines indicated. A more specific indication is given by the trailing hexadecimal byte value. When converted to binary this will map onto the exact lines in error (a bit 'set' is a line in error a bit 'reset' is a correct line).

Copy ZIF Low Data Bus <byte> - Inappropriate levels have been found present on the lines indicated. A more specific indication is given by the trailing hexadecimal byte value. When converted to binary this will map onto the exact lines in error (a bit 'set' is a line in error a bit 'reset' is a correct line).

Copy ZIF Output Enable HIGH/LOW - Voltage level applied lies outside the required tolerance (the direction of the error being indicated by HIGH/LOW).

Copy Zif Vcc HIGH/LOW - Vcc supply level lies without the required tolerance (the direction of the error being indicated by HIGH/LOW).

Disable XON/XOFF for Binary Formats - During attempted use of a binary transmission format, through the serial port, software handshaking was enabled.

ERROR: No Module Connected - When system was powered up no module was found to be connected. If module was in place during power up re-check for correct module insertion (see section 3.3 'Insertion/Removal of Modules').

Erasing Error - A device in a copy ZIF socket has failed to be erased. Any device which has failed to be erased will be marked by flashing LED ZIF status indicators (see section 5.7 'Appendix G LED ZIF Status Indicators').

Failed to Identify Device - During a programming attempt the system tests expected the programmable device to identify itself but it failed to do so. For those devices that have fail the LED ZIF status indicators will now be flashing (see section 5.7 'Appendix G LED ZIF Status Indicators').

FRAMING ERROR - During an attempt to receive, via the serial port, one or more stop bits were expected but not received (see sections 4.7.5 'Stop Bits' and 4.7.4 'Data Bits').

ILLEGAL ACCESS - An attempt was made to unlock system memory with the incorrect access code (see section 4.5.5 'Unlock Memory').

Invalid Character Received - A non-ASCII character has been received whilst attempting communication with an external device using an ASCII based data transfer format (see section 5.2 'Appendix B Data Transfer Formats').

Invalid Format - A data transfer format other than that currently selected on the XR16 system has been received during attempted external communication (see sections 4.7.1 'Transfer Format' and 5.2 'Data Transfer Formats').

Master ZIF Chip Enable HIGH/LOW - Voltage level applied lies outside the required tolerance (the direction of the error being indicated by HIGH/LOW).

Master ZIF Empty - No device has been found in the master ZIF. If a device is in place the error may be caused by an incorrect device being currently selected or by a dirty ZIF (see sections 4.4 'Device Selection' and 5.8 'Appendix H Cleaning ZIFs').

Master ZIF Output Enable HIGH/LOW - Voltage level applied lies outside the required tolerance (the direction of the error being indicated by HIGH/LOW).

Master ZIF Vcc HIGH/LOW - Vcc supply level lies without the required tolerance (the direction of the error being indicated by HIGH/LOW).

NO RAM FOUND ! - During system start-up checks no RAM was found on the system.

Not Permitted - Memory Locked! - Whilst the system RAM space was locked an attempt was made to modify its contents (see sections 4.5.4 'Lock Memory' and 4.5.5 'Unlock Memory').

PARITY ERROR - During an attempt to receive, through the serial port, whilst parity checking was enabled, incorrect data was received (see section 4.7.6 'Parity').

Port Error - Internal check has discovered a hardware port control fault.

POWER SUPPLY ERROR - The voltage levels output from the power supply have been found to lie outside the required tolerance by an internal check.

Programming Voltage HIGH/LOW - Voltage level applied lies outside the required tolerance (the direction of the error being indicated by HIGH/LOW).

PROM ERROR - A device has been placed incorrectly in a ZIF socket. Please check with device orientation symbol located on the right-hand side of your module. This error may also be caused by the insertion of a faulty device or by dirty ZIF sockets (see section 5.8 'Appendix H Cleaning ZIFs').

Ram Overflow - The volume of data from an external transmitter caused an overflow of system RAM.

RAM OVERFLOW - An overflow of system RAM occurred during memory modification.

Ready to Receive: Start Tx - A communications link has been established between the XR16 system and some external transmitter but as yet no data has been received.

Ready to Transmit: Start Rx - A communications link has been established between the XR16 and some external receiver which is at present unready to receive data.

ROM FAULT - Internally programmed checksum value does not equal that calculated during recent check.

Serial Buffer Overflow - During serial communication the XR16 received sufficient data to overflow its input buffer. If handshaking of any form is enabled this should not occur and the transmitter is making incorrect use of the handshaking.

Set 8 Data Bits for Binary Formats - All binary transmission formats require the full 8-bits of data. An attempt was made to transmit using 7 data bits (see sections 4.7.4 'Data Bits' and 5.2 'Appendix B Data Transfer Formats').

SP RAM FAULT - Test reveals that the scratch pad RAM is faulty.

ZIF High Address Lines (A15-A8) <byte> - Inappropriate levels have been found present on the address lines indicated. A more specific indication is given by the trailing hexadecimal byte value. When converted to binary this will map onto the exact lines in error (a bit 'set' is a line in error a bit 'reset' is a correct line).

ZIF High Address Lines (A19-A16) <byte> - Inappropriate levels have been found present on the address lines indicated. A more specific indication is given by the trailing hexadecimal byte value. When converted to binary this will map onto the exact lines in error (a bit 'set' is a line in error a bit 'reset' is a correct line).

ZIF Low Address Lines (A7-A0) <byte> - Inappropriate levels have been found present on the address lines indicated. A more specific indication is given by the trailing hexadecimal byte value. When converted to binary this will map onto the exact lines in error (a bit 'set' is a line in error a bit 'reset' is a correct line).

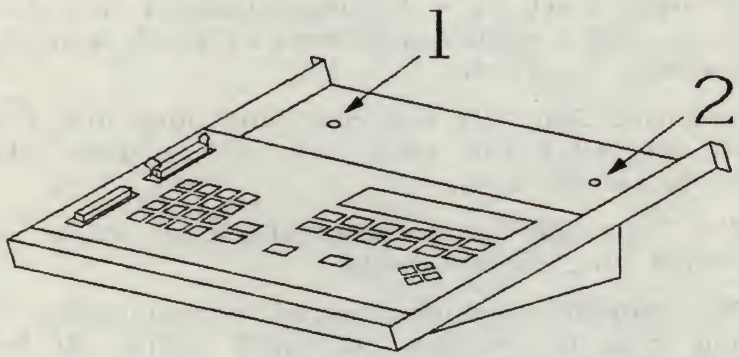
5.5 Appendix E Software Updates

All software updates will arrive in the form of preprogrammed devices. When you receive such a device you will need to insert it into the correct socket within the XR16 mainframe unit.

Before attempting to update the software ensure that the mainframe unit has been switched off and disconnected from the mains.

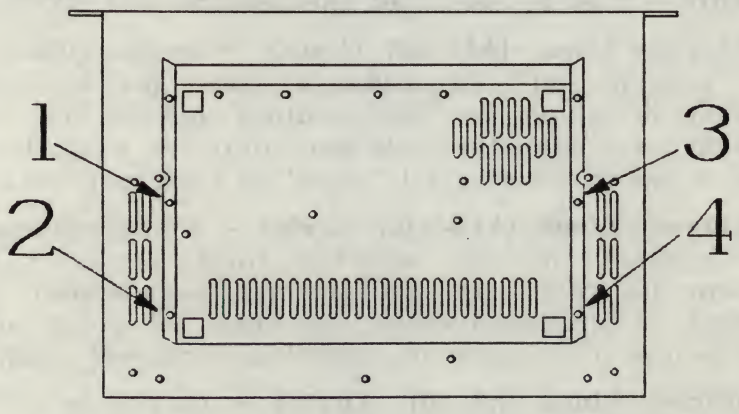
The first step to make this replacement is to remove any module you may have in the XR16 mainframe (see section 3.3 'Insertion/Removal of Modules').

The removal of the module will reveal two screws. Both of these screws must now be removed (see diagram that follows).



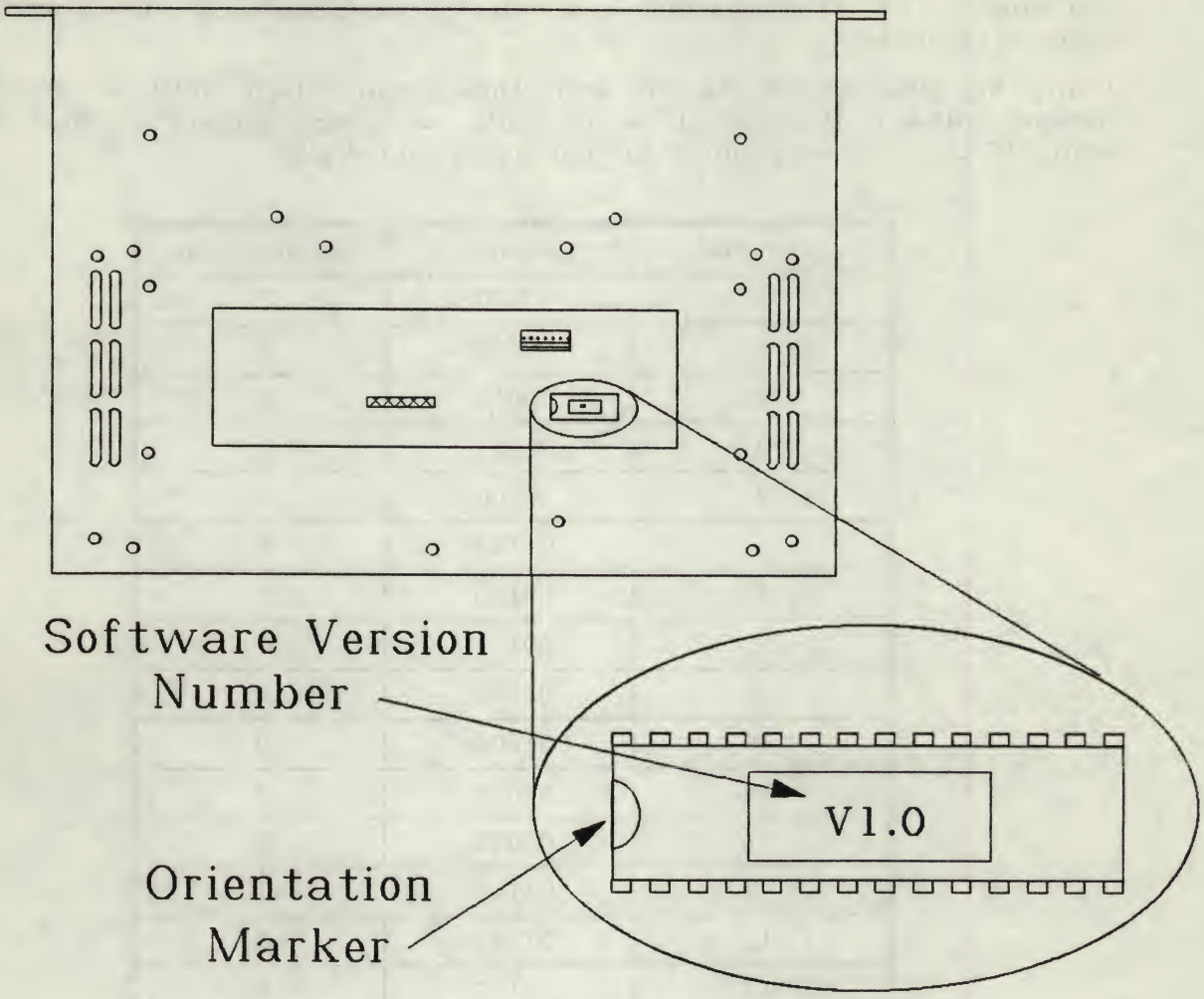
The Upper-side of the XR16

Once these have been removed turn over the XR16 mainframe unit and place it face down on the desk. Four more screws must now be remove each of which are numbered and pin-pointed in the diagram that follows.



The Under-side of the XR16

When all the screws have been removed carefully withdraw the base of the XR16 unit. Two braid cables connect the power supply to the processor board. Disconnect both at the processor board end and detach the base of the XR16.
Locate the device to be replaced, indicated on the diagram that follows, and remove it from its socket.



Software ROM Location

Insert the new device in the socket vacated and ensure the orientation marker faces in the direction indicated in the diagram. Re-connect the two braid leads and re-attach the XR16 base. The software update procedure is now complete.

Software updates for the XR16 will be made available periodically. For information concerning the latest updates please contact our sales office giving details of your current version number.

5.6 Appendix F Hexadecimal Notation

Hexadecimal notation is the most commonly used numeric notation used when dealing with computers and micro processors. It is actually number base 16 just as denary is number base 10 and binary number base 2.

When representing decimal values a single digit may be used until we reach the number 10. At this point, and subsequent powers of 10, an additional digit is required.

Using hexadecimal we do not need this second digit until we reach the decimal number 16. Also it is not until we reach powers of the decimal value 16 that we we require further additional digits.

Decimal	Binary	Hexadecimal
0	00000	0
1	00001	1
2	00010	2
3	00011	3
4	00100	4
5	00101	5
6	00110	6
7	00111	7
8	01000	8
9	01001	9
10	01010	A
11	01011	B
12	01100	C
13	01101	D
14	01110	E
15	01111	F
16	10000	10

5.7 Appendix G LED ZIF Status Indicators

The LEDs below the ZIFs in the XR16 module are used to indicate the status of the devices in the ZIFs (see section 2.1 'View of XR16').

The master ZIF LED is used to indicate whether power is applied to the devices in the ZIFs.

* - **LED on** indicates the power has been applied to ZIFs.
* - **LED off** indicates that all the ZIFs are powered down.

The copy LEDs are used to indicate the results of operations on the ZIFs. There is no relationship between copy LEDs and the power up/down condition of the ZIFs.

* - **LED on** for a particular copy ZIF is on then that device has passed whatever test has been carried out (e.g. Verify).
* - **LED flashing** for any copy ZIF flashing indicates that this device has failed whatever test was being carried out.
* - **LED off** for any copy ZIF indicates that this socket is either empty or there is no appropriate information available about that socket at the present time.

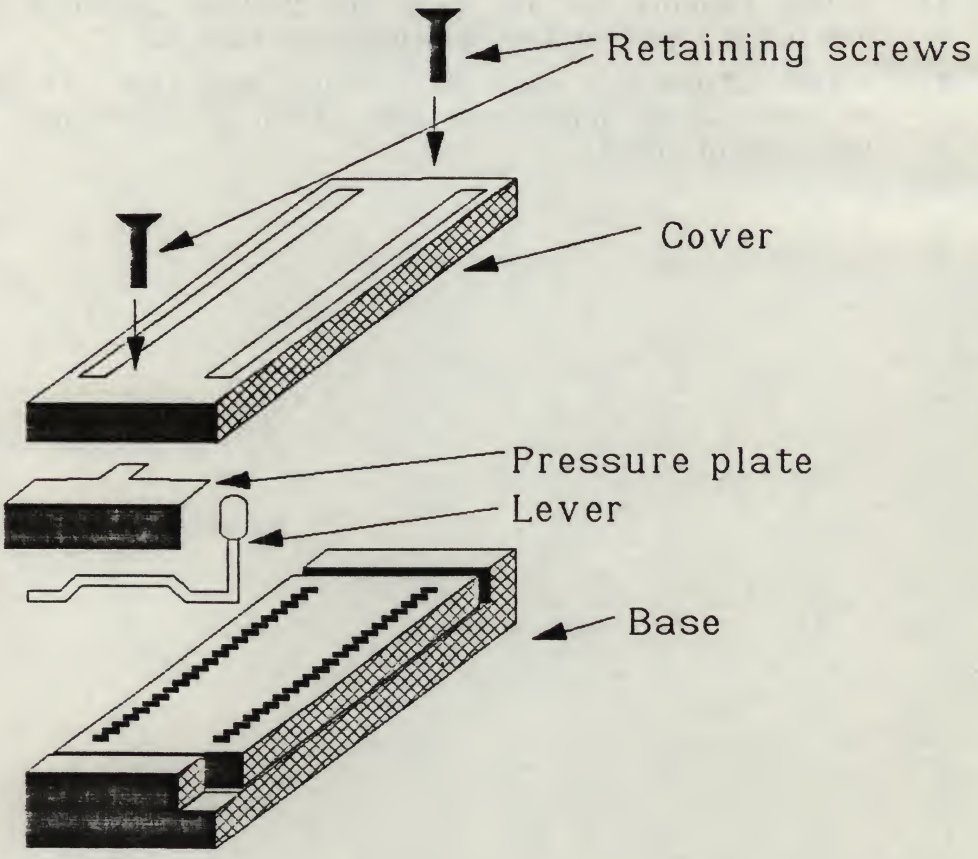
5.8 Appendix H Cleaning ZIFs

In order to clean ZIF sockets with retaining screws the following procedure should be executed:

* Remove the two cover retaining screws.
* Remove the top of the ZIF, taking care to avoid loss of the ZIF lever or pressure plate.
* Examine the contacts carefully. Common contaminants include metallic dust, grease and oil.
* Brush vigorously with a bristle brush sprayed with acetone.
* Reassemble carefully, not forgetting to include the pressure plate.

If your module ZIFs have no retaining screws the following procedure should be executed:

* Brush contacts vigorously with a bristle brush sprayed with acetone whilst the ZIF lever is in the open position.



5.9 Appendix I Modules Available for the XR16

R10 Module

Supporting the gang programming of eight 24/28/32-pin EPROM/EEPROMs simultaneously.

R20 Module

Supporting the gang programming of sixteen 24/28/32-pin EPROM/EEPROMs simultaneously.

R30 Module

Supporting the gang programming of eight 40-pin EPROMs of up to 4-Mbit capacity simultaneously.

R40 Module

Supporting the gang programming of sixteen 40-pin EPROMs of up to 4-Mbit capacity simultaneously.

Also available soon:

XRDRIVE

Software package to communicate between XR16 and 'IBM PC' and true compatibles. Features: supports batch processing (including loops and run-time comments) - makes full use of colour monitors - enables the down-load of data onto floppy or hard disk - can be used to carry out all XR16 operations from the PC without inhibiting its operation.

For further information concerning any of the above contact our sale office stating area of interest. Telephone (0752) 342961

Vcc Verify Function

In order to test programmed device operation over a wider range of Vcc levels, a two pass verify operation has been implimented with the Vcc voltage set at 5v - 5% and 5v + 5%. This function will test the device operation over the manufacturers specified operating voltage range.

XR16 Segmented Data Transfers.

The XR16 can accept segmented data from the RS232 port in any one of the extended address formats (e.g Intel HEX 02, Motorola S2 or Extended Tektronix). When the XR16 is set up for Serial Input with one of these formats selected, a prompt for OFFSET followed by a prompt for SEGMENT is given.

The OFFSET address is added to the address of the data defined by the incoming records, to give the absolute address into which the data will be loaded. The offset required to load data to Ram address XXXXX is given by XXXXX + the 2's compliment of the data address.

The SEGMENT number defines which 512K byte data segment will be downloaded from the incoming file, any data targeted outside this segment range will be ignored by the programmer. The segment number is calculated as:-

Segment = (Data address + offset)/segment length

where the segment length = 512K bytes (80000H)

i.e segment number = (data address + offset address) shifted right 19 bits.

SEGMENT Number (entered in response to SEGMENT prompt.)	Corresponding Data Address Range (from where data will will be loaded.)
00	000000 - 07FFFF
01	080000 - 0FFFFF
02	100000 - 17FFFF
03	180000 - 1FFFFF
.	.
.	.
.	.
.	.
1E	F00000 - F7FFFF
1F	F80000 - FFFFFF

The table shown above shows the correspondence between the XR16 segment number and the valid target address range of the incomming data.

Note that the SEGMENT number defines a 512K byte address range (minimum XR16 RAM size) and may not correspond directly to the segments defined by the various extended data transfer formats.

Example.

An Intel Hex file has been generated which contains data "hello" to reside at address 4C000H in a target system. This data is required to be loaded into the XR16 ram at address 1F000.

A corresponding Extended INTEL HEX file would be:-

```
:020000024000BC
:05C000006865C6C6F27
:00000001FF
```

The UPPER SEGMENT BASE ADDRESS (USBA) as defined by the extended address record	4000H
The USBA shifted left 4 bits.	40000H
Add the record address	C000H
Target address of data	4C000H

For a RAM address of 1F000H, the offset required is 1F000H + 2's compliment 4C000H = D3000H.

The Absolute load address is the data address + offset, 4C000H + D3000H = 11F000H.

The absolute address 11F000H lies within the XR16 segment 2 address range (11F000H >> 19 = 02H).

i.e the file shown above will be loaded into the XR16 ram at address 1F000 by specifying an offset address of D3000H and a load segment of 2 with the Intel 02 format selected.

XR16 Manual Supplement (Remote Mode).

XR16 Remote Mode (Software versions 1.9/3.9 and above).
In response to customer feedback the remote mode on the XR16 was modified commencing at the above software versions. The following provides documentation of the new remote mode operation.

Command Entry
The XR16 sends a prompt ">" to indicate that it requires a command. If the command is not recognized by the XR16 it will prompt with an additional ">". Commands must be entered in upper case.

Display of Zif Status
The XR16 sends its zif status as a string preceded by a ":". The string consists of an eight digit string of characters. The characters are a hexadecimal representation of two 16 bit masks which present the zif status. The first mask has bits set to a one for pass or a zero for fail/unoccupied. The second mask has bits set to a one to indicate which zif is are occupied. The least significant bit of the masks represents zif 1 and the most significant represents zif 16. For an 8 zif module the top eight bits of both masks will always be zero.

Entry of Numeric Data
The XR16 sends a "?" to indicate that it requires data input. If data is entered that is not acceptable to the XR16 it will reply with a "*" to indicate that the data should be re-entered.

Error Handling
The XR16 sends a "!" to indicate an error, this is followed by a single digit error code as shown in the error table. Some of these errors are then followed by a number which gives more specific information on the type of error. These numbers are explained in more depth in the main XR16 manual.

Example
Sequence of events to perform a checksum on the RAM.

XR16 sends	**:00000000>**	No devices in zifs and the XR16 is ready to accept a command.
Type	**RA**	Select RAM to operate on.
XR16 sends	**:00000000>**	Command accepted.
Type	**CS**	Issue checksum command.
XR16 sends	**?**	Request for RAM start address.
Type	**0000**<Return>	Enter RAM start address. Note that just typing enter would use the defaults (see main XR16 Manual).
XR16 sends	**?**	Request for ROM start address.
Type	**10000**<Return>	Enter incorrect ROM start address.
XR16 sends	*****	The data was out of range.
Type	<Return>	Use default ROM start address (0000).
XR16 sends	**?**	Request for length parameter.
Type	**1000**<Return>	Enter length of 1000H.

After a short pause the XR16 sends:

XR16 sends	**:00020002>**	Zif 2 is occupied and has passed a verify against the RAM.

It should be noted that although the XR16 only sends a "?" to prompt for data entry in remote mode, the order of data entry is exactly the same as in local operation. Therefore the main XR16 manual provides guidance on how to use the machine in remote mode.

Device Selection

Device selection differs between local and remote mode. Remote device selection is initally a little more complex, but is much faster in regular use. There are three functions used in device selection:

Query Manufacturer

The "QM" command will return a list of possible manufacturers. The first item in the list is manafacturer No 1, the second No 2 etc.

Query Device

The "QD" command will return a list of devices which correspond to the selected manafacturer.

When QD is entered the XR16 will prompt with a "?". This is a request for a manufacturer number. A number should be entered which corresponds to one of the manufacturer numbers obtained using "QM".

Set Device

The "SD" command is used to select a device and manufacturer. It prompts with a "?"; the manufacturer number should be typed (in hexadecimal) in response to this. When the two digit code for the manufacturer has been entered the XR16 will again prompt with a "?". The device code should be entered in response to this prompt. Once a device has been correctly selected the XR16 will respond with the ">" prompt. To check the device selection the "ST" command should be used.

XR16 Remote Commands

Command	Function
"BI"	BITS PER WORD FOR PROGRAMMING ETC.
"BL"	BLANK CHECK
"CH"	CHAIN PROGRAM
"CO"	COPY
"CR"	CYCLIC REDUNDANCY CHECK
"CS"	CHECKSUM
"EL"	ENTER LABEL DATA FOR PRINTING
"ER"	ERASE
"FI"	FILL
"HL"	DATA IN RAM TREATED AS NORMAL (HI/LO BYTES NOT REVERSED)
"IB"	ILLEGAL BIT CHECK
"ID"	INTELLIGENT IDENTIFIER ENABLED
"IN"	INVERT
"IO"	INTELLIGENT IDENTIFIER DISABLED
"LH"	DATA IN RAM REVERSED (HI/LO BYTES REVERSED)
"LO"	LOCK/UNLOCK THE SYSTEM
"MD"	MEMORY DUMP (80H BYTES FROM ADDRESS GIVEN)
"ME"	MEMORY EDITOR
"PI"	PARALLEL IN
"PF"	PARALLEL FORMAT
"PO"	PARALLEL OUT

Error Code	XR16 Message
O	"Invalid Character Received"
P	"Ram Overflow"
Q	"POWER SUPPLY ERROR"
R	"PROM ERROR"
S	"Programming Voltage"
T	"PORT ERROR"
U	"PARITY ERROR"
V	"RAM paging error"
W	"NO RAM FOUND !"
X	"Master ZIF Output Enable"
Y	"Master ZIF Chip Enable"
Z	"ERROR: No Module Connected"
a	"Master ZIF Vcc"
b	"Master Zif Empty"
c	"Not Permitted - Memory Locked!"
d	"Failed to Identify Device"
e	"Internal Error"
f	"FRAMING ERROR"
g	"Erasing Error"
h	"Copy ZIF Output Enable"
i	"Copy ZIF Chip Enable"
j	"Copy ZIF Vcc"
k	"Copy Zifs Empty"
l	"Checksum Mismatch"
m	"Cannot Erase this device"
n	"ILLEGAL ACCESS"
o	"Illegal Bit Check Failed"

Command	Function
"PP"	PRINT LABELS TO PARALLEL PORT
"PR"	PROGRAM
"QD"	RETURNS DEVICES FOR A GIVEN MANUFACTURER NO.
"QF"	RETURNS LIST OF POSSIBLE DATA TRANSMISSION FORMATS
"QM"	RETURNS LIST OF MANUFACTURERS
"RA"	OPERATE FROM RAM
"RE"	READ E2PROM
"RO"	OPERATE FROM ROM
"RP"	REPLACE
"SA"	SAVE PARAMETERS TO E2PROM
"SD"	SELECT DEVICE
"SE"	SEARCH
"SI"	SERIAL IN
"SF"	SERIAL FORMAT
"SO"	SERIAL OUT
"SP"	PRINT LABELS TO SERIAL PORT
"SR"	STORE DATA FROM MASTER DEVICE
"ST"	STATUS DISPLAY
"VE"	VERIFY
"VI"	VISUAL VERIFY

XR16 Remote Mode Error Codes.

Error Code	XR16 Message
A	"ZIF Low Address Lines (A7-A0)"
B	"ZIF High Address Lines (A15-A8)"
C	"ZIF High Address Lines (A19-A16)"
D	"Copy ZIF Low Data Bus"
E	"Copy ZIF High Data Bus"
F	"Zif VPP Driver"
G	"Block can't be copied onto itself"
H	"Disable XON/OFF for Binary Formats"
I	"Can't send XON"
J	"Can't send XOFF"
K	"Invalid Format"
L	"SERIAL BUFFER OVERFLOW"
N	"SP RAM FAULT"

XR16 Production Mode

1/11/89

With effect from software version 5.8 the XR16 has a new mode. This mode is a production lock. It is accessed from the **PARAMS** menu. The name of the mode is **PROLOCK.**

Description of Function:

The function locks the XR16 into a production mode, once locked the only functions available are those in the **Funcs** menu. In production mode the functions will only work from a master device, not from RAM. The XR16 remains in production mode even when powered down, ensuring that the machine may be used in a production environment without any worries about accidental mis-setting of device types etc.

Operation of Prolock:

When the **Prolock** key is pressed the XR16 will prompt for a production lock code. The only code which may be used is **BACC.** Once this code is entered the XR16 will warn that it is in production mode, offering the chance to **Cont**inue or **Unlock.** The continue option will produce the **Funcs** menu. The Unlock option will request a code to unlock the machine, the code is **BACC.**

Exiting from Prolock:

Pressing **Stop** will display the **Cont Unlock** prompts. Selecting **Unlock** and entering the **BACC** code will deselect the **Prolock** mode.

Additional Points:

1/. When **Prolock** is selected all of the currently selected parameters will be saved under user #1. This will of course overwrite any parameters currently saved under user #1.

2/. Exiting from **Prolock** will always unlock the Editor lock.

3/. The **Lock** function has been renamed **Edlock** but works exactly as previously.

The Prolock code is **BACC**

Index

List of Programmable Devices for the XR16 High Performance Production Programming System Nov 1989

Array Size and Technology	Prog Voltage	Device part number	Module Required
ADVANCED MICRO DEVICES (AMD)			
2048X8 NMOS	25V	Am 9716DC	R10/R20
2048X8 NMOS	25V	Am 2716	R10/R20
2048X8 NMOS	25V	Am 2716B	R10/R20
2048X8 NMOS	25V	Am 2817A	R10/R20
4096X8 NMOS	25V	Am 2732	R10/R20
4096X8 NMOS	25V	Am 9732	R10/R20
4096X8 NMOS	21V	Am 2732A	R10/R20
4096X8 NMOS	12.5V	Am 2732B	R10/R20
4096X8 NMOS	21V	Am 9732A	R10/R20
8192X8 NMOS	21V	Am 2764	R10/R20
8192X8 NMOS	21V	Am 2764 intelligent	R10/R20
8192X8 NMOS	21V	Am 9764	R10/R20
8192X8 NMOS	12.5V	Am 2764A	R10/R20
8192X8 NMOS	12.5V	Am P2764A	R10/R20
8192X8 EEPROM	5V	Am 2864A	R10/R20
8192X8 EEPROM	5V	Am 2864B	R10/R20
16384X8 NMOS	21V	Am 27128	R10/R20
16384X8 NMOS	12.5V	Am 27128A	R10/R20
16384X8 NMOS	12.5V	Am P27128A	R10/R20
32768X8 NMOS	12.5V	Am 27256DC	R10/R20
32768X8 CMOS	12.5V	Am 27C256DC	R10/R20
32768X8 CMOS	12.5V	Am P27C256DC	R10/R20
32768X8 NMOS	12.5V	Am P27256DC	R10/R20
65536X8 NMOS	12.5V	Am 27512	R10/R20
65536X8 CMOS	12.5V	Am 27C512	R10/R20
65536X8 CMOS	12.5V	Am P27C512	R10/R20
65536X16 CMOS	12.5V	Am 27C1024 40 pin	R30/R40
131072X8 CMOS	12.5V	Am 27C010	R10/R20
ATMEL			
512X8 CMOS	5V	28C04	R10/R20
2048X8 CMOS	5V	28C16	R10/R20
2048X8 CMOS	5V	28C17	R10/R20
8192X8 CMOS	5V	28C64	R10/R20
32768X8 CMOS	12.5V	27C256	R10/R20
65536X8 CMOS	12.5V	27C512	R10/R20
4X16384X8 CMOS	12.5V	27C513	R10/R20
2X32768X8 CMOS	12.5V	27C515	R10/R20
EUROTECHNIQUE			
2048X8	25V	ET 2716	R10/R20
4096X8	25V	ET 2732	R10/R20
8192X8	21V	ET 2764	R10/R20
EXEL			
2048X8	5V	2816A	R10/R20
2048X8	5V	2817A	R10/R20
8192X8	5V	2864A	R10/R20
8192X8	5V	2865A	R10/R20
FAIRCHILD SEMICONDUCTOR			
2048X8 NMOS	25V	F 2716	R10/R20
FUJITSU			
2048X8 NMOS	25V	MBM 8516	R10/R20
4096X8 NMOS	25V	MBM 2532	R10/R20
4096X8 NMOS	25V	MBM 2732	R10/R20
4096X8 NMOS	21V	MBM 2732A	R10/R20
8192X8 NMOS	21V	MBM 2764	R10/R20
8192X8 NMOS	21V	MBM 2764 intelligent	R10/R20
8192X8 CMOS	21V	MBM 27C64	R10/R20
16384X8 NMOS	21V	MBM 27128	R10/R20
32768X8 NMOS	12.5V	MBM 27256	R10/R20
32768X8 CMOS	21V	MBM 27C256	R10/R20
32768X8 CMOS	12.5V	MBM 27C256A	R10/R20
65536X8 CMOS	12.5V	MBM 27C512	R10/R20
131072X8 CMOS	12.5V	MBM 27C1000 32 pin	R10/R20
131072X8 CMOS	12.5V	MBM 27C1001 32 pin	R10/R20
65536X16 CMOS	12.5V	MBM 27C1024 40 pin	R30/R40
GENERAL INSTRUMENT (GI)			
32768X8 NMOS	12.5V	27256	R10/R20
32768X8 CMOS	12.5V	27C256	R10/R20
HITACHI			
2048X8 NMOS	21V	HN 462716/G	R10/R20
2048X8 EEPROM	25V	HN 48016P	R10/R20
4096X8 NMOS	25V	HN 462532	R10/R20
4096X8 NMOS	25V	HN 462732	R10/R20
4096X8 NMOS	21V	HN 462732A	R10/R20
8192X8 NMOS	21V	HN 482764	R10/R20
8192X8 NMOS	21V	HN 482764 intelligent	R10/R20

Array Size and Technology	Prog Voltage	Device part number	Module Required
HITACHI			
8192X8 NMOS	5V	HN 58064	R10/R20
8192X8 CMOS	5V	HN 58C65	R10/R20
16384X8 NMOS	21V	HN 4827128	R10/R20
32768X8 NMOS	12.5V	HN 27256	R10/R20
32768X8 CMOS	12.5V	HN 27C256	R10/R20
65536X8 NMOS	12.5V	HN 27512	R10/R20
131072X8 CMOS	12.5V	HN 27C101 32 pin	R10/R20
131072X8 CMOS	12.5V	HN 27C301 32 pin	R10/R20
INTEL			
1024X8 NMOS	25V	2758A	R10/R20
1024X8 NMOS	25V	2758B	R10/R20
2048X8 NMOS	25V	2716	R10/R20
2048X8 EEPROM	5V	2816A	R10/R20
2048X8 EEPROM	5V	2816B	R10/R20
2048X8 EEPROM	5V	2817A	R10/R20
4096X8 NMOS	25V	2732	R10/R20
4096X8 NMOS	21V	2732A	R10/R20
8192X8 NMOS	21V	2764	R10/R20
8192X8 NMOS	21V	2764 intelligent	R10/R20
8192X8 NMOS	12.5V	2764A	R10/R20
8192X8 NMOS	12.5V	P2764A	R10/R20
8192X8 CMOS	12.5V	27C64	R10/R20
8192X8 CMOS	12.5V	P27C64	R10/R20
8192X8	5V	2864A	R10/R20
8192X8	5V	2864B	R10/R20
16384X8 NMOS	21V	27128	R10/R20
16384X8 NMOS	12.5V	27128A	R10/R20
16384X8 NMOS	12.5V	P27128A	R10/R20
16384X8 NMOS	12.5V	27128B	R10/R20
32768X8 NMOS	12.5V	27256	R10/R20
32768X8 NMOS	12.5V	P27256	R10/R20
32768X8 CMOS	12.5V	27C256	R10/R20
32768X8 CMOS	12.5V	P27C256	R10/R20
32768X8 CMOS		27F256	R10/R20
65536X8 NMOS	12.5V	27512	R10/R20
65536X8 NMOS	12.5V	P27512	R10/R20
4X16384X8 NMOS	12.5V	27513	R10/R20
4X16384X8 NMOS	12.5V	P27513	R10/R20
128KX8 HMOS	12.5V	27010 32 pin	R10/R20
8X16KX8 HMOS	12.5V	27011 28 pin	R10/R20
65536X16 HMOS	12.5V	27210 40 pin	R30/R40

Array Size and Technology	Prog Voltage	Device part number	Module Required
MITSUBISHI			
2048X8 NMOS	25V	M5L 2716K	R10/R20
4096X8 NMOS	25V	M5L 2732K	R10/R20
4096X8 NMOS	21V	M5L 2732A	R10/R20
8192X8 NMOS	21V	M5L 2764K	R10/R20
8192X8 NMOS	21V	M5L 2764 intelligent	R10/R20
16384X8 NMOS	21V	M5L 27128	R10/R20
16384X8 CMOS	21V	M5L 27C128	R10/R20
32768X8 NMOS	12.5V	M5L 27256	R10/R20
65536X8 NMOS	12.5V	M5L 27512	R10/R20
131072X8 CMOS	12.5V	M5L 27C100 32 pin	R10/R20
131072X8 CMOS	12.5V	M5L 27C101 32 pin	R10/R20
65536X16 CMOS	12.5V	M5L 27C102 40 pin	R30/R40
MOSTEK			
2048X8 NMOS	25V	MK 2716	R10/R20
8192X8 NMOS	21V	MK 2764	R10/R20
8192X8 NMOS	21V	MK 2764 intelligent	R10/R20
MOTOROLA			
2048X8 NMOS	25V	MCM 2716	R10/R20
4096X8 NMOS	25V	MCM 2532	R10/R20
4096X8 NMOS	25V	MCM 68732-0	R10/R20
4096X8 NMOS	25V	MCM 68732-1	R10/R20
8192X8 NMOS	25V	MCM 68764	R10/R20
8192X8 NMOS	25V	MCM 68766	R10/R20
NATIONAL SEMICONDUCTORS			
1024X8 NMOS	25V	MM 2758A	R10/R20
1024X8 NMOS	25V	MM 2758B	R10/R20
2048X8 NMOS	25V	MM 2716	R10/R20
2048X8 CMOS	25V	NMC 27C16	R10/R20
2048X8 EEPROM	5V	MM 9817	R10/R20
2048X8 EEPROM	25V	MM 6716	R10/R20
4096X8 NMOS	25V	MM 2532	R10/R20
4096X8 NMOS	25V	MM 2732	R10/R20
4096X8 CMOS	25V	NMC 27C32	R10/R20
8192X8 NMOS	21V	MM 2764	R10/R20
8192X8 NMOS	21V	MM 2764 intelligent	R10/R20
8192XR CMOS	12.5V	NMC 27C64	R10/R20
8192X8 EEPROM	5V	NMC 98C64	R10/R20
16384X8 CMOS	12.5V	NMC 27C128	R10/R20
16384X8 CMOS	12.5V	NMC 27CP128	R10/R20
32768X8 CMOS	12.5V	NMC 27C256	R10/R20
65536X8 CMOS	12.5V	NMC 27C512	R10/R20

Array Size and Technology	Prog Voltage	Device part number	Module Required
NIPPON ELECTRIC (NEC)			
1024X8 NMOS	25V	uPD 2758	R10/R20
2048X8 NMOS	25V	uPD 2716	R10/R20
4096X8 NMOS	25V	uPD 2732	R10/R20
4096X8 NMOS	21V	uPD 2732A	R10/R20
8192X8 NMOS	21V	uPD 2764	R10/R20
8192X8 NMOS	21V	uPD 2764 intelligent	R10/R20
8192X8 CMOS		uPD 27C64	R10/R20
16384X8 NMOS	21V	uPD 27128	R10/R20
32768X8 NMOS	21V	uPD 27256	R10/R20
32768X8 CMOS	21V	uPD 27C256	R10/R20
32768X8 NMOS	12.5V	uPD 27256A	R10/R20
32768X8 CMOS	12.5V	uPD 27C256A	R10/R20
65536X8 CMOS	12.5V	uPD 27C512	R10/R20
131072X8 CMOS	12.5V	uPD 27C1000 32 pin	R10/R20
131072X8 CMOS	12.5V	uPD 27C1001 32 pin	R10/R20
65536X16 CMOS	12.5V	uPD 27C1024 40 pin	R30/R40
262144X8 CMOS	12.5V	uPD 27C2001 32 pin	R10/R20
O.K.I. ELECTRIC			
1024X8 NMOS	25V	MSM 2758	R10/R20
2048X8 NMOS	25V	MSM 2716	R10/R20
4096X8 NMOS	25V	MSM 2532	R10/R20
4096X8 NMOS	25V	MSM 2732	R10/R20
4096X8 NMOS	21V	MSM 2732A	R10/R20
8192X8 NMOS	21V	MSM 2764	R10/R20
8192X8 NMOS	21V	MSM 2764 intelligent	R10/R20
131072X8 NMOS	21V	MSM 27128	R10/R20
RICOH			
4096X8 NMOS	21V	2732A	R10/R20
ROCKWELL			
2048X8 EEPROM	5V	R 2816	R10/R20
2048X8 EEPROM	5V	R 5213	R10/R20
2048X8 EEPROM	5V	R 52B13	R10/R20
4096X8 CMOS	25V	R 2732	R10/R20
4096X8 CMOS	21V	R 87C32	R10/R20
8192X8 EEPROM	5V	R 52B33	R10/R20
8192X8 NMOS	21V	R 2764	R10/R20
8192X8 CMOS	21V	R 87C64	R10/R20

Array Size and Technology	Prog Voltage	Device part number	Module Required
SEEQ			
8192X8 NMOS	5V	DQ2864	R10/R20
8192X8 NMOS	5V	DQ2864H	R10/R20
8192X8 NMOS	5V	DQ28C64	R10/R20
8192X8 NMOS	5V	52B33	R10/R20
8192X8 NMOS	5V	55B33/H	R10/R20
8192X8 NMOS	5V	52B33H	R10/R20
2048X8	5V	5213/H	R10/R20
2048X8	5V	52B13/H	R10/R20
8192X8 NMOS	21V	DQ2764	R10/R20
8192X8 NMOS	21V	DQ2764 intelligent	R10/R20
16384X8 NMOS	21V	DQ27128	R10/R20
32768X8 CMOS	5V	28C256	R10/R20
32768X8 CMOS	5V	27C256	R10/R20
2048X8 NMOS	5V	2816A	R10/R20
2048X8 NMOS	5V	2816H	R10/R20
2048X8 NMOS	5V	5516A	R10/R20
2048X8 NMOS	5V	2817A	R10/R20
2048X8 NMOS	5V	2817AH	R10/R20
2048X8 NMOS	5V	5517A	R10/R20
2048X8 NMOS	5V	5517AH	R10/R20
SGS ATES			
2048X8 NMOS	25V	M 2716	R10/R20
2048X8 NMOS	25V	P2716	R10/R20
4096X8 NMOS	25V	M 2532	R10/R20
4096X8 NMOS	25V	M 2732	R10/R20
4096X8 NMOS	21V	M 2732A	R10/R20
8192X8 NMOS	21V	M 2764 intelligent	R10/R20
8192X8 NMOS	12.5V	M 2764A	R10/R20
16384X8 NMOS	12.5V	M 27128A	R10/R20
32768X8 NMOS	12.5V	M 27256	R10/R20
65536X8 NMOS	12.5V	M 27512	R10/R20
SIGNETICS			
8192X8 CMOS	12.5V	27C64A	R10/R20
32768X8 CMOS	12.5V	27C256	R10/R20
SYNERTEC			
2048X8 NMOS	25V	2716	R10/R20

Array Size and Technology	Prog Voltage	Device part number	Module Required
TEXAS INSTRUMENTS			
1024X8 NMOS	25V	TMS 2508	R10/R20
2048X8 NMOS	25V	TMS 2516	R10/R20
4096X8 NMOS	25V	TMS 2532	R10/R20
4096X8 NMOS	21V	TMS 2532A	R10/R20
4096X8 NMOS	25V	TMS 25L32	R10/R20
4096X8 NMOS	25V	TMS 2732	R10/R20
4096X8 NMOS	21V	TMS 2732A	R10/R20
4096X8 NMOS	21V	TMS P2732A	R10/R20
8192X8 NMOS	25V	TMS 2564	R10/R20
8192X8 NMOS	21V	TMS 2764	R10/R20
8192X8 NMOS	21V	TMS 2764 intelligent	R10/R20
8192X8 NMOS	21V	TMS P2764	R10/R20
16384X8 NMOS	21V	TMS 27128	R10/R20
16384X8 CMOS	12.5V	TMS 27C128	R10/R20
32768X8 CMOS	12.5V	TMS 27C256	R10/R20
32768X8 NMOS	12.5V	TMS 27256	R10/R20
65536X8 CMOS	12.5V	TMS 27C512	R10/R20
THOMSON-EFC			
4096X8 NMOS	25V	EF 2532	R10/R20
8192X8 CMOS	12.5V	EF 27C64	R10/R20
32768X8 CMOS	12.5V	EF 27C256	R10/R20
TOSHIBA			
2048X8 NMOS	25V	TMM 323	R10/R20
4096X8 NMOS	25V	TMM 2732	R10/R20
4096X8 NMOS	21V	TMM 2732A	R10/R20
8192X8 NMOS	25V	TMM 2564	R10/R20
8192X8 NMOS	21V	TMM 2764	R10/R20
8192X8 NMOS	21V	TMM 2764 intelligent	R10/R20
16384X8 NMOS	21V	TMM 27128	R10/R20
16384X8 NMOS	12.5V	TMM 27128A	R10/R20
32768X8 NMOS	21V	TMM 27256	R10/R20
32768X8 CMOS	21V	TC 57256	R10/R20
32768X8 CMOS	12.5V	TC 57256A	R10/R20
65536X8 NMOS	12.5V	TMM 27512	R10/R20
131072X8 CMOS	12.5V	TC 571000 32 pin	R10/R20
131072X8 CMOS	12.5V	TC 571001 32 pin	R10/R20

Array Size and Technology	Prog Voltage	Device part number	Module Required
XICOR			
512X8 EEPROM	5V	X 2804A	R10/R20
2048X8 EEPROM	5V	X 2816A	R10/R20
2048X8 EEPROM	5V	X 2816B	R10/R20
8192X8 EEPROM	5V	X 2864A	R10/R20
8192X8 EEPROM	5V	X 2864B	R10/R20
8192X8 EEPROM	5V	X 28C64	R10/R20
8192X8 EEPROM	5V	X 2864H	R10/R20
32768X8 EEPROM	5V	X 28256	R10/R20
32768X8 EEPROM	5V	X 28C256	R10/R20

This Table was compiled from manufacturers data and is correct to the best of our Knowledge

GP Industrial Electronics Ltd.,
Unit E, Huxley Close,
Newnham Ind. Estate,
Plymouth PL7 4JN

Tel: (0752) 342961
Telex: 42513 SHARET
Fax: (0752) 342764

GP Industrial Electronics Ltd.

Unit E. Huxley Close. Newnham Industrial Estate. Plymouth PL7 4JN
Telephone: (0752) 342961 Telex: 42513 SHARET G

NEW DEVICES ADDED TO XR16 ,V6.1

NEC	27C4001
NAT SEMI	27C1024
TOSHIBA	27256A
INTEL	28F256(P1),28F256(P2)
INTEL	28F512
INTEL	28F010
TEXAS	27C32
TEXAS	27C64
INTEL	87C64

CHANGE TO XR16 SERIAL/PARALLEL IN

10/5/90. XR16 Version 6.2 and greater.

Serial in no longer prompts for a segment address for formatted serial/parallel input. Instead the OFFSET address has been extended to a maximum of six bytes, and this value is now SUBTRACTED from the record address.

For example, if you have a file starting with an address of 18000 HEX and you wish it to load into the XR16 at address zero, then enter an offset of 18000 HEX. If you want to load it at 8000 HEX enter an offset of 10000 HEX.

Any data which has a resultant address which lies outside the XR16 RAM will be ignored.

Warranty: One year

For a period of 1 year from the date of purchase, GP Industrial Electronics Ltd warrants this product to be free from defects in Material or Workmanship.
This warranty applies only to the original purchaser who purchased the unit from GP Industrial Electronics Ltd or its duly authorised agents and dealers. The warranty shall be voided if the unit has been subjected to improper or abnormal use, or if the unit is altered, modified or any attempt is made to repair. If a defect occurs during the warranty period, the unit must be returned to GP Industrial Electronics Ltd or the agent from whom the unit was purchased for repair, along with proof and date of purchase, and a detailed description of any faults.
Cost of shipment to and from GP Industrial Electronics Ltd's facility shall be bourne on account of the buyer.

GP Industrial Electronics Ltd's sole responsibility in the event of defect is limited to the correction of defects by adjustment, repair, or replacement at GP Industrial Electronics Ltd's sole expense and discretion.

Other than repair, there are no warranties express or implied, included but not limited to, any implied warranties or merchantability or for particular applications. In no event shall GP Industrial Electronics Lts be liable for loss of profits or benefits or other similar damages arising out of any breach of this warranty or otherwise.

.....

Software Updates

Any software updates that GP Industrial Electronics Ltd may introduce, whether designed to rectify bugs in the software, or to enchance the machines capabilities shall be provided free of charge during the 1 year period from the date of purchase of the machine, on the condition that the original set of EPROMs supplied with the machine or a set of blank EPROMs to be programmed are supplied to GP prior to updating.
Alternatively GP will require a payment in advance to cover the cost of media.

For non UK customers, please contact your local dealer who will advise you on software updates.